Apuntes resumidos Sistemas de la...

user120723

Sistemas de Información

3º Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

SISTEMAS DE LA INFORMACIÓN

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

Que te den **10 € para gastar** es una fantasía. ING lo hace realidad.

Abre la **Cuenta NoCuenta** con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Quiero el cash

Consulta condiciones **aquí**

do your thing

Sistemas de Información

Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

1. INTRODUCCIÓN

1.1. INTRODUCCIÓN

Datos: valores en crudo que representan hechos.
Información: colección de datos organizados de forma que proporcionen valor añadido. Los datos se convierten en información al definir relaciones entre ellos de forma que resulten útiles.
Conocimiento: conciencia o familiaridad adquirida por la experiencia de hechos o situaciones a través del aprendizaje, observación o introspección.

El origen de los datos se denomina **provenance** y la **trazabilidad** o **data lineage** indica los procesos a los que han sido sometidos.

Proceso: conjunto de tareas lógicamente relacionadas que a partir de datos de entrada proporciona resultados (datos de salida)
Algoritmo: Lista ordenada de pasos o especificación de instrucciones para llevar a cabo una determinada tarea. Aplicando un algoritmo a los datos se puede obtener información.
Sistema: conjunto de elementos (procesos, algoritmos, conocimiento, información y datos) que interactúan para lograr un objetivo.

Los **componentes** de cualquier tipo de sistema son:

- **Entradas**
- **Mecanismos de procesado**: elementos físicos de procesamientos y otros elementos como protocolos.
- **Salidas**
- **Feedback**

Las **métricas** de cualquier tipo de sistema son:

- **Feedback**
  - **Efectividad**: mide en qué medida se ha alcanzado el objetivo del sistema.
  - **Eficiencia**: mide el beneficio con respecto al consumo realizado para obtenerlo (optimización de recursos).

- **Medidas de rendimiento estándar** específicas del sistema.

1.2. SISTEMA DE INFORMACIÓN

Es un **sistema** que está compuesto por un conjunto de **elementos interrelacionados** que **recogen** (entrada), **manipulan** (proceso), **almacenan** información, y **diseminan** (salida) datos y, además, proporcionan **mecanismos correctores** (feedback) para alcanzar un determinado objetivo. Definición de R.Stair & G.Reynolds.

Componentes de un SI:

- **Hardware**: ordenadores, infraestructura de comunicación y redes.
- **Software**: sistemas y aplicaciones.
- **Datos**: configuración y específicos del sistema.
- **Personas**: usuarios no técnicos y técnicos.
- **Procedimientos y protocolos**: políticas de uso, reglas de mantenimiento y revisión, control de acceso, metodología, implantación y control de calidad, etc.

Clasificación de los SI: (Stair y Reynolds)

- **Sistemas de procesado transaccional (TPS)**: gestionan la información referente a las transacciones producidas diariamente en una organización, como la compra de materiales, registro de horas de los empleados, ventas de productos, etc.
- **SI de gestión (MIS)**: orientados a los responsables técnicos de las diferentes áreas de la organización para la definición de procedimientos rutinarios. Son SI operativa: generación de nóminas, facturación, asignación de tareas, etc.
- **Sistemas de apoyo a la toma de decisiones (DSS)**: dan soporte a la toma de decisiones para un problema específico complejo. Además, en general la información a considerar para analizar el problema no está definida.
- **SI para ejecutivos (EIS)**: DSS para altos directivos. Orientados a conseguir los objetivos estratégicos de la empresa. Su principal uso es informativo.

1.3. APLICACIONES EMPRESARIALES

CARACTERÍSTICAS

- **Almacenan y manipulan datos**: Bases de datos y ficheros XML (intercambio de datos y configuraciones)
- **Realizan transacciones**: propiedades ACID (Atomicity-Consistency-Isolation-Durability)
- **Escalables**: más carga de trabajo sin necesidad de modificar el software.
- **Disponibles**: no dejan de prestar servicio.
- **Seguras**: permisos acceso a datos y funcionalidades.
- **Integración**: diferentes tecnologías.

Hay distintos tipos de **interfaces**:

- Texto
- Entorno de ventanas
- Web
- Aplicaciones móviles

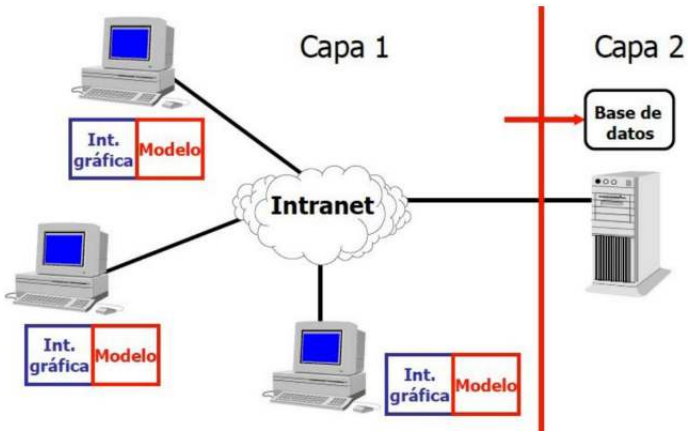
ARQUITECTURAS SOFTWARE

Aplicaciones monocapa:

- **Modelo de datos**: dependiente de la aplicación en concreto (no tiene en cuenta la integración con el resto del sistema/aplicaciones)
- **Persistencia**: ficheros.
- **Ventajas**: rápidas, útiles para aplicaciones de propósito específico.
- **Inconvenientes**:
  - Necesaria instalación y re-compilación en todas las máquinas.
  - Datos duplicados y necesidad de procesos ETL.

Aplicaciones de dos capas

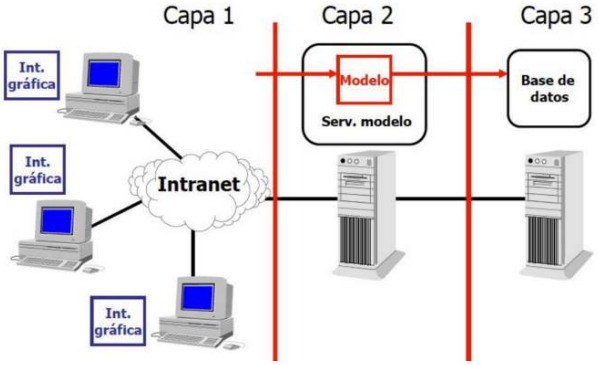
- **Separación entre interfaz y modelo**:
  - **Modelo**: clases que implementan las reglas de negocio y que son independientes de la interfaz.
  - **Interfaz**: clases que afectan a la navegación de la aplicación y a la visualización de los datos.
- **Ventajas**:
  - Cada capa puede ser desarrollada por personal con perfiles específicos.
  - Reuso de la capa modelo para diferentes dispositivos.



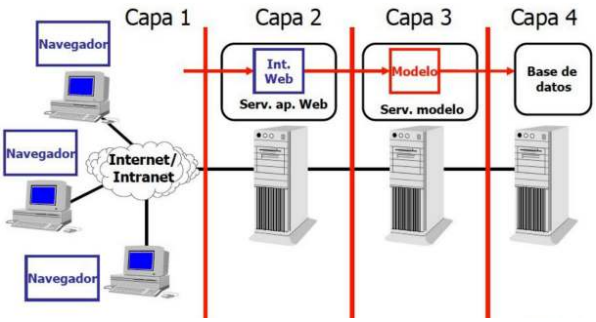
- **Inconvenientes**:
  - Cambios en el modelo implican la re-compilación e instalación en todas las máquinas cliente.

Aplicaciones de tres capas:

- **Ventajas**:
  - Cambios en el modelo sólo afecta al servidor de la aplicación.
  - Clientes ligeros que necesitan poca capacidad de procesamiento
- **Inconvenientes**:
  - Cambios en la interfaz gráfica implican la re-compilación y reinstalación de la aplicación cliente.



Aplicaciones de cuatro capas:
Esta arquitectura suele emplearse cuando la interfaz gráfica web y la capa modelo están construidas con tecnologías diferentes. Se requiere una aplicación Web para facilitar el acceso a la aplicación.



TECNOLOGÍAS

- **Acceso a bases de datos**: API JDBC(Java DataBase Connectivity) o API ODBC (Open DataBase Connectivity)
- **Aplicaciones Web**:
  - servlets y JSP para la interfaz y Java EE para el modelo.
  - ASP.Net para la interfaz y C# y ADO.Net para el modelo.
  - PHP Ruby on Rails, Python, MEAN, etc.
- **Servidores Web**: Tomcat, Jboss, Jetty,WebSphere.

1.4. NUEVAS TENDENCIAS: CLOUD COMPUTING

El **cloud computing** es un paradigma que permite el **acceso ubicuo bajo demanda** a servicios TIC a través de internet. Surge de la **externalización del servicio TIC y para ahorrar costes**.

TIPOS DE CLOUD

Según la **funcionalidad**:

- **SaaS** (Software as a Service) → Google Apps:
  - Se ofrecen aplicaciones en concreto sin controlar el cliente ni la infraestructura hardware ni su configuración.
  - SaaS = PaaS + despliegue de aplicaciones.
  - Características:
    - No es necesaria inversión en licencias ni en mantenimiento.
    - Dependencia de la red y el proveedor.
    - Aspectos legales y de seguridad.
    - Los clientes comparten infraestructura HW Y SW aunque cada uno tenga su propio espacio.
- **PaaS** (Platform as a Service) → MarketPlaces:
  - Se ofrecen entornos de desarrollo cooperativo y despliegue de aplicaciones rápido.
  - PaaS=IaaS + middleware, herramientas de desarrollo, servicios de inteligencia empresarial, SOs, sistemas de administración de bases de datos, etc.
  - Diseñado para sustentar el ciclo de vida completo de las aplicaciones Web: compilación, pruebas, implementación, administración y actualización.
  - Características: No es necesaria demasiada inversión en administración de sistemas.

- **IaaS** (Infraestructure as a Service) → AWS de Amazon:
  - Ofrece recursos de computación: almacenamiento, red, procesadores, etc.
  - Características:
    - Recursos compartidos y virtualización.
    - Gestión más eficiente de los data-centers.
    - No es necesaria una gran inversión inicial.
    - Ahorro de costes de mantenimiento. electricidad, etc.
    - Aumento de la seguridad.

Según la **compartición**:

- **Público**: cloud destinado a público general o empresas que deseen contratarlo. → Amazon Web Services.
- **Privado**: para uso exclusivo de una organización. Puede ser propio o alquilado. En las máquinas donde se ejecutan los sistemas de la empresa que alquila no se ejecutan sistemas de otras empresas. → Inditex.
- **Híbrido**: Los picos se gestionan mediante un cloud público de forma totalmente transparente al usuario. → Amazon Web Services.
- **Comunitario**: cloud privado de una comunidad de organizaciones.

2. EVOLUCIÓN DE LA WEB

2.1. TIPOS DE FUENTES DE DATOS

Fuentes de datos no estructurada:

- Documentos o audio en lenguaje natural cuyo contenido es interpretado por personas.
- Búsqueda de información en repositorios de documentos cerrados, la Web o interfaces basadas en palabras clave

Fuentes de datos estructuradas:

- Datos que tienen asociados un conjunto de metadatos.
- Búsqueda de información localizando la fuente de datos y analizando su estructura (metadatos) o en interfaces basadas en SQL.

Fuentes de datos híbridas (La Web)

2.2 WEB ESTÁTICA O TRADICIONAL (Web 1.0)

Internet: nodos interconectados y conjunto de protocolos que permite comunicar y compartir datos e información entre nodos y compartir recursos. El origen fue el proyecto **DARPA** en la guerra fría y **ARPANET**.

La **Web** fue un proyecto impulsado por Tim Berners Lee a finales de los 80. Su objetivo era facilitar la compartición e intercambio de documentos entre los diferentes científicos involucrados en un proyecto. Tiene 3 elementos clave: **HTML** (HyperText Markup Language), **URL** (Uniform Resource Locator), **HTTP** (HyperText Transfer Protocol).

HTTP es un protocolo sin estado que sigue un esquema petición-recurso. Los recursos se identifican mediante URL. Las peticiones pueden ser de tipo GET (los parámetros se codifican en la URL), POST (los parámetros se codifican en el cuerpo del mensaje), HEAD, PUT, DELETE. Las respuestas del servidor incluyen un código de estado:

- 1xx (información)
- 2xx (éxito)
- 3xx (redirección)
- 4xx (error cliente)
- 5xx (error servidor)

Las URL son un tipo de URI (Unified Resource Identifier). Su formato es: Protocolo://([usuario:clave@]dominio o ip[:puerto])[/]ruta acceso [?parámetros] [#identificador de fragmento]].

CARACTERÍSTICAS

Definida como **Web solo lectura** en la que la mayoría de los usuarios visualizan páginas mediante navegadores. La **búsqueda de información** se realizaba a través de portales y directorios de búsqueda. El **mayor problema** es que la información generalmente está desactualizada ya que la actualización supone un coste alto.

ATAQUES Y AMENAZAS

Los ataques **externos** se producían contra la disponibilidad, la integridad, la confidencialidad y la conservación. Los ataques **internos** son el factor humano o el fallo técnico provocando un contenido de la Web obsoleto o interfaces pobres.

2.3. WEB DINÁMICA O DE TRANSICIÓN (Web 1.5)

CARACTERÍSTICAS

Se produce un **cambio tecnológico** respecto a la Web 1.0 en el que los documentos HTML se generan de forma dinámica cuando se solicitan, accediendo a bases de datos o ficheros, evitando así información y datos obsoletos. Se diferencia la ejecución en cliente vs. servidor. La Web sigue siendo una Web de **solo lectura**.

Sesión: periodo de tiempo durante el cual un determinado usuario interactúa con un sitio Web. La debe implementar cada aplicación o usar alternativas como incluir el id de sesión en las cookies, transmitir el id de sesión en las URL o incluir el id de sesión en los campos ocultos del formulario.

La **búsqueda de información** se hace mediante motores de búsqueda (indexado automático).

ATAQUES Y AMENAZAS

El **factor humano** o **fallo técnico** como realizar la comprobación de errores en los parámetros solo en los clientes, *SQL injection*, usar solo https en la autenticación inicial y después http (el id de sesión viaja en claro) o Cross Site Scripting.

2.4. WEB SOCIAL (2.0)

CARACTERÍSTICAS

Se produce un **cambio de filosofía y uso**. Se usan formularios para la recogida de datos e información que se incluirán en el sitio Web y las aplicaciones se centran en los usuarios que participan en la elaboración de contenido. Por lo que se convierte en una **Web de lectura/escritura**. Ejemplos: wikipedia, redes sociales, etc.

La **búsqueda de información** pasa a ser más compleja por la cantidad de sitios web y páginas, por lo que surge la necesidad de escalar sistemas de búsqueda. También hay buscadores específicos como Google Scholar.

ATAQUES, AMENAZAS Y VULNERABILIDADES

Se crea una necesidad de contrastar la información y las fuentes de datos. Además se dificulta la localización de contenido. Surge la **Deep Web**.

2.5. WEB SEMÁNTICA (Web 3.0)

CARACTERÍSTICAS

Esta Web está orientada a máquinas y personas en la que los agentes SW pueden interactuar de forma automática con distintos sitios Web. Es una **Web de lectura, escritura y ejecución**. Supone una **revolución tecnológica** basada en tres pilares fundamentales: anotación de recursos con metadatos, ontologías comunes y uso de reglas y razonadores para deducir nuevos datos que permitan tomar decisiones.

ANOTACIÓN SEMÁNTICA

Se basa en la existencia de metadatos que identifican los datos. Para realizar anotaciones e integrar información semántica en las páginas web se usa **RDFa** y para recuperar información semántica, **SPARQL**.

ONTOLOGÍAS

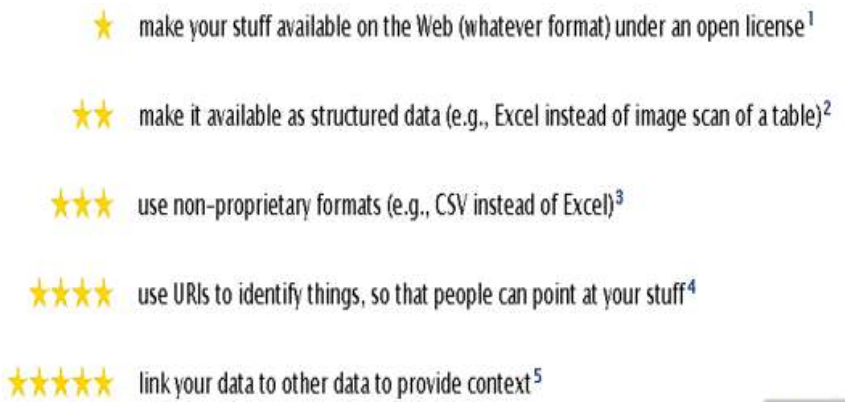
Se crea un vocabulario estandarizado común. Para definirlas se emplea **RDF** (Resource Description Framework), **RDFS** (RDF Schema) y **OWL** (Ontology Web Language).

REGLAS Y RAZONADORES

Hay motores de inferencia automáticos y motores de razonamiento (razonadores).

LA WEB DE DATOS ENLAZADOS (LINKED DATA)

Para que los datos enlazados sean óptimos hay que cumplir los siguientes principios:



La aplicación que demostró la viabilidad de la Web Semántica fue **DBpedia**, cuyo objetivo es hacer accesible a los agentes software la información disponible en las cajas de información de Wikipedia.

Hay tres reglas para el Linked Data:

- Definir una **URI** para cada **entidad** o recurso.
- **Formato estándar** al mostrar información de URIs mediante http: URI "deferenciables".
- **Relacionar los datos** entre sí.

3. TECNOLOGÍAS DE LA WEB ESTÁTICA

3.1. HTML: HYPER TEXT MARKUP LANGUAGE

Lenguaje basado en **etiquetas** que indican los diferentes tipos de elementos. Las etiquetas se escriben entre los símbolos **<** y **>**. Cada etiqueta tiene una marca que indica su comienzo y su final y pueden tener atributos → *<etiqueta atrib1="valor"* >contenido*</etiqueta>*. Pueden existir etiquetas sin contenido → *<etiqueta></etiqueta>* o de forma abreviada *<etiqueta/>*.

HTML5

Aparecen nuevas características destinadas a la creación de **aplicaciones Web** y no solo a la creación de documentos HTML (fichero de texto escrito en HTML que los navegadores interpretan). Se produce una separación entre contenido y la forma de mostrarlo (CSS). Se integra el tratamiento de contenido multimedia y los gráficos vectoriales (etiqueta *<canvas>*). Un fichero HTML se puede editar en un simple editor de texto, en entornos de programación, en entornos específicos (Adobe Dreamweaver) u otras opciones.

ESTRUCTURA DE UN DOCUMENTO HTML

Cabecera (título, estilo y metainformación) y **cuerpo** (contenido).
Comentarios: *<!--Comentarios pueden ser multi-línea-->*

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Ejemplo</title>
<meta name="description"content="indefinido"/>
</head>
<body>
    <p>¡Hola Mundo!</p>
</body>
</html>
```

ETIQUETAS DEL ENCABEZADO

<title>Titulo***</title>***
<style>Definiciciones del estilo de la página***</style>***
<script>Definiciones de JavaScript para dar dinamismo a la página***</script>***
<link rel="StyleSheet href="estilo.css" type="text/css/>
<meta name="elemento a describir"***content***="descripión"***/>***

ETIQUETAS DE TEXTO

Párrafos: *** … ***
Encabezados con 6 niveles de encabezamiento: ***<h1></h1>,…,<h6></h6>***
Cambio de línea: ***
***
Resaltar texto: ***……***

ETIQUETAS DE IMÁGENES

******

ETIQUETAS DE ENLACES

*******Contenido que genera la peticion al clicar*******

ETIQUETAS PARA PROPORCIONAR INFORMACIÓN EN FORMA DE TABLA

<table>contenido de la tabla***</tabla>***
Definir encabezado: ***<th>***Nombre del encabezado***</th>***
Definir filas: ***<tr>***contenido de la fila***</tr>***
Definir columnas: ***<td>***contenido de la columna***</td>***

Ejemplo:
```
<body><table>
    <tr><th>Alumno</th><th>Nota</th></tr>
    <tr><td>Pepe Camino</td><td>9</td></tr>
    <tr><td>Ana Yus</td><td>8</td></tr>
</table></body>
```

ETIQUETAS PARA PROPORCIONAR LISTADOS

Listas no ordenadas: ***
item 1
item N
***
Listas ordenadas: ***
item 1
item N
***
Listas de definiciones: ***<dl><dt>***término a definir***</dt><dd>***definición***</dd></dl>***

ETIQUETAS PARA CAPTURAR INFORMACIÓN EN EL CLIENTE

La etiqueta ***<form>*** captura información en el cliente para proporcionársela al servidor en una nueva petición de recurso.
La etiqueta ***<input>*** captura información.

Ejemplo:
```
<form name="registro" action="/procesarForm.do" method="get">
    <label for="campo1"> Nombre: </label>
    <input type="text" name="nombre" id="campo1"/>
    <label for="campo2">Apellidos: </label>
    <input type="text" name="apellidos" id="campo2"/>
    <label for="campo3">Clave: </label>
    <input type="password" name="clave" id="campo3"/>
</form>
```

La etiqueta input permite seleccionar una opción entre múltiples posibilidades: ***type="radio"***
Ejemplo: <*input type="radio" name*="sex" *value*="H"/>
<*input type="radio" name*="sex" *value*="M"/>

También permite la selección de múltiples valores ***type="checkbox"***
Ejemplo: <*input type="checkbox" name*="vehiculo" *value*="coche"/>
<*input type="checkbox" name*="vehiculo" *value*="moto"/>
<*input type="checkbox" name*="vehículo" *value*="bici"/>

Botón de envío: ***<input type="submit"/>***
Botón de reset: ***<input type="reset/>***
Otros botones: ***<input type="button" onclick=***"llamadaAFunción"***/>***
Campos ocultos: ***<input type="hidden" name=***"variableX" ***value=***"unValor"***/>***

Etiqueta ***<textarea>***: textos de mayor tamaño que los campos input
Listas desplegables: ***<select>*** y ***<option>*** (multivaluadas). Ejemplo:

```
<select name="Marca_Vehículo">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```

CREAR SECCIONES EN EL DOCUMENTO

La etiqueta *** *** define secciones en el documento. Se le pueden aplicar estilos de presentación diferenciados.

OTRAS ETIQUETAS DE MULTIMEDIA

Sonido:
<audio src="ruta al fichero que se desea reproducir" ***preload="none, auto o metadata" controls autoplay loop></audio>***

<audio preload autoplay controls loop>
<source src="ruta fichero principal en formato ogg" ***type="audio/ogg>***
<source src="ruta fichero alternativo en formato mp3" ***type="audio/mpeg"***
</audio>

Video:
<video controls>
<source src="ruta fichero principal en formato ogg" ***type="video/ogg>***
<source src="ruta fichero alternativo en formato mp4" ***type="video/mp4" codecs="avc1.42E01E,mp4a.40.2">***
</video>

3.2. HOJAS DE ESTILO CSS (CASCADING STYLE SHEETS)

INTRODUCCIÓN CSS

Asociadas al lenguaje HTML para definir el **formato** de presentación de los elementos del documento. Se interpretan igual que el lenguaje HTML en el cliente web. Se definen en el atributo **style** de cada etiqueta **(inline)** o en la cabecera (<style>) en declaraciones de estilo de la forma: **selector {propiedad:valor;}(interno)**.

TIPOS DE SELECTORES: (EXTERNO)

Básicos: el nombre de una etiqueta. h1{text-align:center;}
De clase: nombre de la clase. efecto1{text-align:center;} <h1 class="efecto1">Header</h1>
De identificador: #nombre del identificador. Similar a los selectores de clase pero para un único elemento de la página. #id1{text-align:center;} <h1 id="id1">Header</h1>

4. TECNOLOGÍAS DE LA WEB DINÁMICA

4.1. VO/DAO

Los **Value Objects** (VO) son implementaciones en memoria (clases y objetos) de las entidades del modelo de aplicación. Corresponden en su mayoría con las tablas de entidades de las bases de datos. La aplicación se comunicará con la base de datos empleando los VO como objetos de transferencia. Un VO debe tener propiedades que representan los atributos del objeto y métodos set y get para acceder a esas propiedades. También se puede añadir un constructor.

Los **Data Access Objects** (DAO) transfieren objetos VO entre la capa de lógica de la aplicación y la capa de persistencia. Estas clases implementan todas las funciones (API) que sean necesarias para "conceptualizar" las necesidades de nuestra aplicación. Además, abstraen y encapsulan la comunicación con la base de datos de manera que la capa de lógica invoca unas funciones en la capa DAO y recibe un resultado.

4.2. SERVLETS Y JSPs

Una **aplicación Web** es una aplicación que se ejecuta en un servidor Web y a la que el usuario accede desde un cliente de propósito general. En ellas hay contenido estático y dinámico.

SERVLET

Un **Servlet** es un ejecutable escrito en lenguaje Java que normalmente se ejecuta en respuesta a una petición http. Puede recibir peticiones HTTP. Procesa datos y genera una respuesta acorde con los parámetros generando una página HTML dinámica. Se ejecutan en diferentes hilos del proceso servidor y son independientes de la plataforma. Cada Servlet puede estar asociado a una o más URLs.

El método público de **HttpServletRequest** denominado **getParameter** permite obtener el valor de un parámetro que tiene asociado un único valor (atributo **monovaluado**). Mientras que el método **getParameterValues** permite obtener el valor de un parámetro **multivaluado** (aunque también se puede emplear con atributos monovaluados).

El método público de **HttpServletResponse setContentType** permite establecer el tipo de contenido de la respuesta y debe llamarse antes de comenzar a escribir en el objeto **PrintWriter** proporcionado por el método **getWriter**.

El método **sendRedirect** le indica al cliente como respuesta a la petición http que ha hecho que genere una nueva petición a la URL que se le indica como parámetro. Es decir, se realiza una redirección con una nueva petición. Sin embargo, también se puede realizar una redirección reenviando la petición actual sin que el cliente sea consciente de ello, para ello se emplea el método **forward**:

```
RequestDispatcher dispatcher = request.getRequestDispatcher(url);
dispatcher.forward(request, response)
```

JAVA SERVER PAGES (JSPs)

Una **página JSP** es un tipo especial de Servlet orientado a crear una interfaz gráfica ya que tiene aspecto de una página HTML y puede incluir scriptlets escritos en Java.

Al acceder a una página JSP:

- Si es la primera vez que se accede a ella, el servidor implementa un Servlet, generado a partir de la página JSP, lo compila y lo carga en memoria.
- Si no es la primera vez, le pasa la petición al servlet asociado.
- Si la página JSP se ha modificado desde la compilación del servlet asociado se genera un nuevo servlet.

Para crear scriptlets se usa <%...%>, para importar clases, <%@...%>, para generar comentarios, <%--...--%> y para incluir expresiones en Java que dan como resultado un String o un objeto convertible a String, <%=expresión%>. Además, las páginas JSP tienen los siguientes elementos declarados de forma implícita:

- **request**: javax.servlet.http.HttpServletRequest
- **response**: javax.servlet.http.HttpServletResponse
- **session**: javax.servlet.http.HttpSession
- **out**: javax.servlet.jsp.JspWriter

CUANDO USAR SERVLETS Y CUANDO JSPs

Los **JSPs** se emplean para la generación de la vista de la aplicación, como la visualización de de formularios y de mensajes de error o la visualización de los resultados de una operación. Por otro lado, los **servlets** se usan para el procesamiento de los formularios y las llamadas a la capa modelo de la aplicación. En el caso del procesamiento de los formularios, si los parámetros son correctos realizará la operación correspondiente y le pasará el control a la página JSP que visualiza los resultados de la acción, mientras que si no lo son detectará los errores y pasará el control a la página JSP que permita al usuario subsanar errores.

EMPAQUETAMIENTO DE UNA APLICACIÓN WEB

Una aplicación Web se puede hacer disponible desplegándola en el servidor Web o mediante ficheros .war. La estructura de este fichero es la siguiente:

- Directorio WEB-INF/classes: contiene los ficheros .class de la aplicación Web agrupados según su estructura de paquetes.
- Directorio WEB-INF/lib: contiene los ficheros .jar de las librerías que usa la aplicación.

Las etiquetas más comunes son:

- **Display-name**: define el nombre de la aplicación Web.
- **Servlet**: declara cada clase servlet que forma parte de la aplicación (**servlet-class**) y le asigna un nombre (**servlet-name**).
- **Servlet-mapping**: define las URLs asociadas a cada servlet (**url-pattern**) definido anteriormente (**servlet-name**).
- **Welcome-file-list**: indica la página devuelta por el servidor cuando se acceda a la aplicación (**welcome-file**).

5. TECNOLOGÍAS DE LA WEB SEMÁNTICA Y WEB DE DATOS

5.1. RDF, RDFS Y SPARQL

RDF es un modelo, representado en forma de grafo, de datos estándar para el intercambio de datos en la Web "extendible" por computadoras y para describir relaciones entre los diferentes "recursos". Un recurso es cualquier concepto del entorno digital o de otro entorno, como una página Web, un dato, un servicio, un libro, articulo, persona, etc.

Está basado en **tripletas** (Sujeto, Predicado, Objeto) que se pueden representar en forma de grafo de forma que los nodos representan los recursos y las aristar dirigidas las relaciones entre los recursos. El sujeto y objeto representan recursos y el predicado una propiedad.

Los recursos y las propiedades se identifican con URIs. Las propiedades deben tener una definición clara y precisa y estar acordadas con una comunidad:

- **Dublin Core**: ontología o vocabulario para recursos bibliográficos.
- **FOAF**: personas y entidades y sus relaciones de amistad y pertenencia.
- **SKOS**: categorías y divisiones de conocimiento.

En RDF se permite un tipo especial de nodo o recurso denominado "**Blank Node**" que puede representar desconocimiento o que no existe una URI que represente a este recurso todavia. Se puede especificar el tipo de dato y el idioma en el que están escritos los valores de las propiedades. El sujeto puede ser URI o Blank Node, el predicado, URI y el objeto, URI, Blank Node o un Literal (valor en concreto de un tipo de dato).

RDFS es un lenguaje basado en RDF para definir vocabularios para RDF (RDF Schema). E sun lenguaje para definir los medatados o la estructura de fuentes de datos RDF.

CLASES

- rdfs:Resource
- rdfs:Class
- rdfs:Literal
- rdfs:DataType
- rdf:XMLLiteral
- rdf:Property

PROPIEDADES

- rdfs:domain
- rdfs:range
- rdf:type
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:label
- rdfs:comment

SPARQL es un lenguaje declarativo para extraer información de grafos RDF. Su funcionamiento se basa en el emparejamineto de patrones de la pregunta contra la Base de Conocimientos que estamos interrigando (**Graph-Pattern_Matching**). La pregunta puede expresarse también en formato de grafo. Los valores que deseamos conocer o son desconocidos los representamos con **variables**: **?nombreDeLaVariable**.

	Modelos relacionales	Modelos basados en tripletas
Componente base	Tablas o relaciones	Tripletas (S, P, O)
Definición de metadatos	Sentencias SQL CREATE TABLE	Tripletas RDF considerando RDFS
Definición de datos o instancias	Sentencias SQL INSERT	Sentencias SPARQL CONSTRUCT o definición de Tripletas RDF
Lenguaje de consulta	Sentencias SQL SELECT	Sentencias SPARQL SELECT y ASK

5.2. SINTAXIS DEL LENGUAJE DE CONSULTA SPARQL

FORMAS DE CONSULTA

SELECT

ASK

- Ejemplo: ASK{
  ?ganador dcterms:subject dbpediaCat:Nobel laurates_in_Literature.
  ?ganador dbpediaprop:birthPlace "España".
  }

CONSTRUCT

- Ejemplo: CONSTRUCT <http://www.unizar.es/Raquel> miVocab:deseoLeer ?ganador{
  ?ganador dcterms:subject dbpediaCat:Nobel laurates_in_Literature .
  }

DESCRIBE

- Ejemplo: DESCRIBE dbpedia:Pablo_Neruda.

MODIFICADORES DE LAS FORMAS DE CONSULTA

ORDER BY: ordena alfanuméricamente las respuestas según el campo que se indique. Por defecto las ordena por orden ascendente.
LIMIT: limita el número de resultados que se muestran.
OFFSET: indica a partir de que resultado se empieza a contar.
OPTIONAL: sirve para indicar que una de las restricciones del WHERE es opcional, de forma que puede no cumplirse. Sirve por ejemplo para añadir un campo que no tengan todas las respuestas, de forma que si un campo la tiene la muestra y si no muestra un espacio en blanco.
FILTER: se filtran los datos en función de las restricciones indicadas. Algunos filtros que se pueden definir son:

- Lang (?variable): especifica el lenguaje en el que está la variable.
- datatype
- regex
- isUri
- isLiteral
- isBlank
- aritméticos (>,<,=,etc)

6. RECUPERACIÓN DE LA INFORMACIÓN

6.1. BÚSQUEDA DE INFORMACIÓN EN CORPUS

INTRODUCCIÓN

El objetivo de los sistemas de Recuperación de Información (**RI**) o Information Retrieval (**IR**) es que dada una colección de documentos (**corpus documental**) y una necesidad de información de un determinado usuario expresada en forma de pregunta (**query**) se recuperen los documentos para resolver la necesidad de información del usuario.

Los componentes básicos de un sistema de RI son:

- Un **formalismo** para representar cada uno de los **documentos**.
- Un **formalismo** para representar las **consultas** (keyword-based interfaces)
- Una **medida de similitud** entre un documento y una consulta.

Hay dos posibles soluciones para medir la similitud: el **matching sintáctico**, para cuando se trabaja con pocos documentos, y el **recorrido secuencial**, en el que se recorre todo el corpus documental comparando el contenido de cada documento con las palabras de la consulta.

El matching sintáctico conlleva ciertos problemas:

- **Polisemia de las palabras**: Aunque aparezcan palabras claves de la consulta en el documento, puede que este no sea relevante para la consulta ya que las palabras se pueden emplear con otro significado al buscado.
- **Sinonimia de las palabras**: aunque no aparezcan ciertas palabras clave, no significa que el documento no sirva para la consulta ya que puede haber sinónimos de dichas palabras.

El recorrido secuencial también:

- Para corpus > 200Mb requiere demasiado tiempo. Para solucionarlo hay varias opciones:
  - **Índices invertidos** para acceder a los documentos.
  - Asociar a cada **término** una **lista de los documentos** donde aparece.
  - **Entradas (elementos del diccionario)**: claves con las que realizar el acceso.
  - **Salidas (listas)**: elementos a los que se desea acceder o recuperar

Para evaluar la relevancia de un sistema de RI hay dos tipos de medidas:

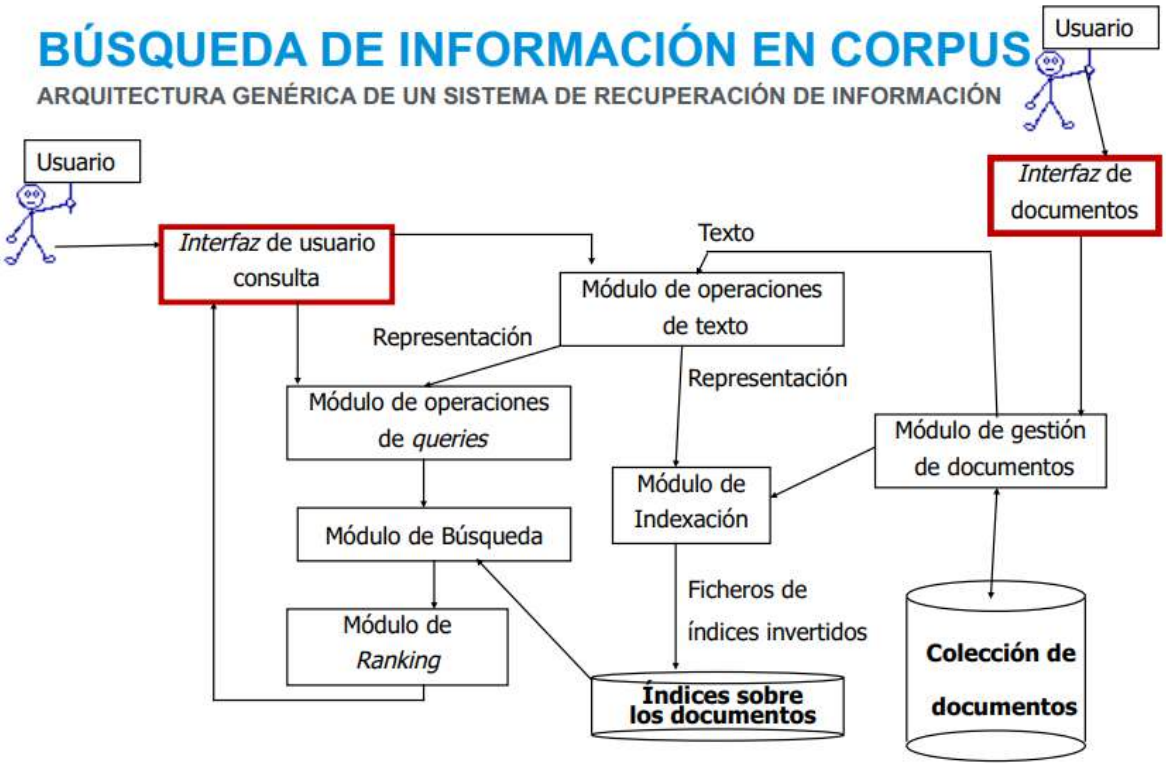
- Medidas de **efectividad**:
  - **Precisión (P)**: nº de doc. relevantes recuperados/nº total de doc. recuperados
  - **Recall (R)**: nº de doc. relevantes recuperados/nº total de dec. relevantes.
  - **F-measurement**: media armónica de P y R: $\{(1+B^2)\ P*R\}/\{B^2*(P+R)\}$
  - **Corpus TREC (1992)**
- Medidas de **eficiencia**: menor cantidad de recursos empleados.

Un índice invertido es una estructura de datos utilizada en sistemas de recuperación de información que asocia cada término del corpus con una lista de documentos donde aparece. En esta lista, puede incluirse información adicional como la frecuencia del término en cada documento o las posiciones exactas. Esto permite realizar búsquedas rápidas y eficientes, especialmente en grandes colecciones de documentos, evitando recorrerlos de forma secuencial.

RI no solo trata modelos para representar documentos y consultas si no también:

- Métodos de almacenamiento de documentos.
- Métodos de compresión para reducir el espacio que ocupa el almacenamiento de los documentos y los índices, distinguiendo entre técnicas de compresión que permiten la búsqueda entre texto comprimido y entre las que no.
- Métodos de indexación de documentos.
- Métodos de presentación (**Ranking**).
- Otros:
    - Clasificación automática de documentos (text-classification).
    - Agrupamiento de documentos similares (clustering)

ARQUITECTURA GENÉRICA DE UN SISTEMA DE RECUPERACIÓN DE INFORMACIÓN



El **módulo de gestión de documentos** se encarga de gestionar los documentos del corpus y los metadatos asociados:

- Insertar, actualizar y/o eliminar un documento del corpus.
- Parsear los documentos que forman la colección para extraer información de ellos.
- Dado un determinado identificador de documento recuperar el documento.
- Dado un determinado identificador de documento consultar los metadatos que tiene asociados.

El **módulo de operaciones de texto** se encarga de transformar el documento/consulta en una representación del mismo/de la misma (vista lógica):

- En general la vista lógica consiste en una secuencia de términos.
- Las técnicas empleadas son: usar listas de palabras sin contenido semántico o lista de *stopwords* (*stoplist*), la **lematización** (*stemming*), que se basa en la consideración de las raíces semánticas de los términos y el procesamiento de lenguaje natural (**NLP**) de nombres compuestos, nombres de entidades o la desambiguación semántica.

El **módulo de indexación** gestiona los índices sobre los documentos. Suele usar los índices invertidos. Se emplea para: determinar la unidad de indexación (el término) y para determinar qué información va a almacenar el índice (lista de identificadores de documentos, posiciones donde aparece el término en el documento o la frecuencia del término).

El **módulo de operaciones de consulta** se emplea para el uso de recursos lingüísticos como Thesaurus Léxicos y ontologías para enriquecer la representación lógica de la consulta. Para ello se usa la expansión de términos de la representación de la consulta (‘coche’ pasa a ser ‘coche o auto o carro’) y la solicitud de retroalimentación al usuario (*relevance-feedback*) para especificar en mayor detalle su consulta (‘manzana’ se refiere a fruta o logotipo de Apple).

El **módulo de operaciones de búsqueda** dada la representación lógica de una query realiza consultas en los índices para determinar cuáles son los documentos más relevantes para dicha consulta. Para ello, la opción más simple es el *matching sintáctico*.

Por último, el **módulo de ranking** ordena los documentos recuperados de acuerdo con la relevancia respecto a la consulta que se está resolviendo. La medida de similitud empleada entre las representaciones lógicas de las consultas y las representaciones lógicas de los documentos puede ser de tres tipos:

- Modelo booleano
- Modelo vectorial
- Modelo probabilístico

Puede tener en cuenta otros aspectos como el contexto en el que se encuentra el usuario y las preguntas anteriores formuladas por el mismo.

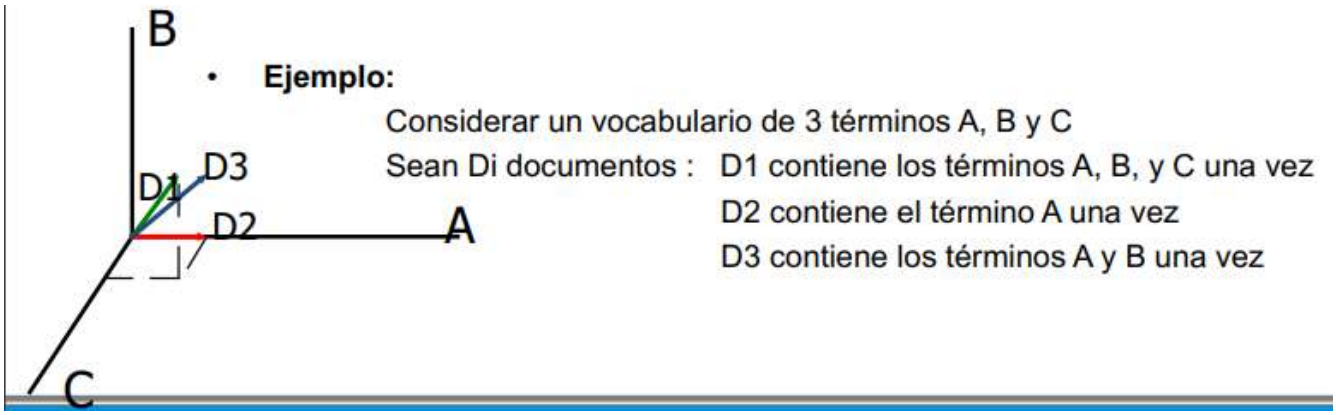
MÓDULO DE BÚSQUEDA Y RANKING

En el **modelo booleano** cada documento se representa por una lista de bits (1 o 0) que indican si en ese documento aparece determinado término del vocabulario del corpus o no. Las ventajas de este modelo es que es sencillo de implementar y rápido debido a que usa operaciones a nivel de bit. Por otro lado, los inconvenientes son el hecho de que puede recuperar o muy pocos o demasiados documentos y el expresar las consultas booleanas resulta complicado al usuario.

Ejemplo: Vocabulario: CASA, SER, VERDE, AZUL, AMARILLA, CARA, BARATA.
Documento: “Mi casa es amarilla y barata” -> (1,1,0,0,1,0,1)
Query 1: Casa barata -> (1,0,0,0,0,0,1)
Query 2: Casa and (amarilla or azul) -> (1,0,0,1,0,0,0) or (1,0,0,0,1,0,0)

En el **modelo booleano extendido** en lugar de considerar solo 0 y 1 en los vectores que constituyen las representaciones de los documentos, se considera el número de veces que aparece un término en el documento. Entre sus ventajas destaca el hecho de que se puede hacer un ranking de los documentos recuperados.

En el **modelo vectorial**, que surge del modelo booleano extendido, las queries y los documentos se representan mediante vectores cuya dimensión es la cardinalidad del vocabulario (conjunto de términos considerados).



La query: documentos con A y B => q =<1,1,0>

La **medida de similitud** de dos vectores es el coseno del ángulo que forman. A mayor similitud entre X e Y menor es el ángulo y mayor el coseno.

Ejemplo: q D1 = 1 1 + 1 1 + 0 1 = 2
|q| = sqrt(1^2 + 1^2 + 0^2) = sqrt(2)
|D1| = sqrt(1^2 + 1^2 + 1^2) = sqrt(3)
Cos(q y D1) = 2/(sqrt(2) sqrt(3)) = 2 / (1.414)(1.732) = 2 / 2.449 = 0,81

Las ventajas del modelo vectorial son las siguientes:

- Formalismo matemático sencillo de implementar
- Poco coste computacional
- Librerías que ya nos lo proporcionan (SMART; Lucene)
- Alto rendimiento

Sin embargo, también hay inconvenientes como que trata cada término como independiente y que no tiene en cuenta el tamaño de los documentos.

El **modelo probabilístico** se basa en probabilidades condicionadas y el teorema de Bayes.

6.2. BÚSQUEDA DE INFORMACIÓN EN LA WEB

INTRODUCCIÓN

La diferencia entre Corpus y Web se basa en la necesidad de localizar los documentos con los que se va a trabajar. Para ello se usan 3 técnicas:

- Localización manual (directorios)
- Localización automática (crawlers)
- Localización híbrida (los crawlers localizan y los usuarios clasifican)

Los **directorios de búsqueda** se basan en la organización manual de las páginas en categorías. Unos de sus inconvenientes son la escalabilidad y la definición de la estructura.

Los **motores de búsqueda** son una adaptación de las técnicas de RI en grandes corpues a la Web mediante el uso de crawlers, generalmente con interfaces basadas en palabras clave.

ARQUITECTURA GENÉRICA DE UN MOTOR DE BÚSQUEDA BÁSICO

Es idéntica a la arquitectura de búsqueda en Corpus solo que en lugar de haber una interfaz de documentos hay crawlers.

Los **motores de búsqueda** se basan en una construcción de un gran índice de palabras sobre todos los documentos de la web estática. Las búsquedas se realizan por palabras clave sobre el índice, obteniendo así la granularidad del documento.

META BUSCADORES

Los **meta buscadores** realizan búsquedas en otros buscadores y luego integran sus resultados en tiempo real. Se han utilizado para mejorar la relevancia en buscadores en Internet mediante algoritmos de ponderación.

Entre sus dificultades se encuentran las siguientes:

- Traducción de consultas del formato general al de la fuente (traducción de sintaxis y post-procesados)
- Construcción de 'envoltorios' sobre los buscadores origen.
- Relevancia ponderada de resultados.

Eficiencia de las consultas.

ALGORITMO PAGE-RANK

Los enlaces son considerados como 'citas' de otros documentos. Se asumen como más relevantes los documentos más citados pero también importa quién es el que te cita. Por tanto, una página tiene un **page-rank** alto si tiene muchas páginas que la apuntas o la apuntan páginas con un PageRank alto ("**Hubs**"). El texto en los enlaces se asocia también a la página destino.

ALGORITMO HITS

La relevancia se basa en **hiperenlaces**, de forma que primero se realiza una búsqueda previa sobre un índice pre-construido y luego se aplica un algoritmo iterativo sobre los enlaces entre documentos. Los **hubs** son páginas que enlazan muchas 'páginas buenas' (**autoridades**).

El algoritmo se basa en que cada página (nodo del grafo) comienza con un 'peso de hub' y un 'peso de autoridad'. En cada iteración el 'peso de autoridad' de un nodo se calcula como la suma del 'peso de hub' de la iteración anterior de los nodos que lo apuntan. El 'peso de hub' se calcula como la suma del 'peso de autoridad' de los nodos a los que apunta. Está demostrado que el algoritmo converge.

LA WEB OCULTA (DEEP WEB)

En la **web oculta** se encuentra el contenido de la Web no indexado por los motores de búsqueda y por tanto difícilmente localizable a no ser que se conozca su existencia. Dichos contenidos no se indexan ya que los servidores que los alojan se encuentran aislados, para solucionarlo bastará con proporcionarle la URL del contenido a un motor de búsqueda y porque hay páginas que se generan dinámicamente.

7. BASES DE DATOS DISTRIBUIDAS

7.1 INTRODUCCIÓN

INTRODUCCIÓN

En una base de datos centralizada se guarda toda la información en la misma base de datos, es decir, hay una centralización de los datos. Algunos de sus problemas son el cuello de botella, las latencias, distintas frecuencias de acceso, etc.
Mientras, las bases de datos distribuidas buscan la integración de los datos en lugar de la centralización.

7.2. BASES DE DATOS DISTRIBUIDAS

Un sistema de **BD distribuidas** es una colección de varias BDD que se encuentran lógicamente interrelacionadas y desplegadas sobre una red de ordenadores. Un **gestor de bases de datos distribuidas** (SGBDD) es el software que permite el manejo de sistemas de BDD distribuidas y que hace dicha distribución **transparente** al usuario.

Las BDD no son Sistemas de BD distribuidas, sino un conjunto de BD que pueden comunicarse unas con otras donde no existe un esquema global y que tienen un mecanismo para ver todas las bases de datos como una única base de datos.

VENTAJAS Y DESVENTAJAS

Algunas de las ventajas son:

- **Transparencia de red**: el usuario no debe ser consciente del uso de la red, ni de la localización de los datos y tiene que haber un espacio de nombres independientes de la localización.
- **Transparencia de fragmentación**: el usuario no debe ser consciente de la existencia de varios depósitos de datos.
- **Transparencia de replicación**: el usuario no debe ser consciente de la existencia de varias copias de datos.
- La **distribución** puede ser la organización más natural.
- Mayor **fiabilidad** y **disponibilidad** (puede haber replicación)
- **Autonomía local**.
- Más **eficiencia** al acceder a los datos locales, frente a una aproximación centralizada.
- **Económicamente** mejor.
- **Escalabilidad**: más posibilidades de expansión.
- **Compartición y disponibilidad** de datos (ya que están en una red).

Algunas de las desventajas son:

- Falta experiencia en el diseño.
- **Complejidad**: se suman los problemas de las BD centralizadas más la seguridad en redes, las transacciones distribuidas y el control distribuido.
- **Coste**: HW+SW+comunicaciones.
- **Dificultad de cambio** de BD centralizada a BDD

DISEÑO DE BDD

El **diseño top-down** se basa en el diseño de un esquema global(E/R, Relacional) que posteriormente se **fragmenta** y finalmente, se elaboran esquemas locales y se **asignan** los fragmentos a los esquemas locales.

Un **fragmento** es la unidad a distribuir, que puede ser una parte de una tabla, una tabla entera o un conjunto de tablas. Hay 3 tipos de fragmentación:

- **Horizontal**: basada en encontrar condiciones de selección. A simple forma es partir una tabla horizontalmente.
- **Vertical**: basada en encontrar conjuntos de atributos a proyectar. A simple forma es como dividir la tabla en columnas verticales, cada fragmento puede contener varias columnas.
- **Híbrida**: primero horizontal y después vertical.

A su vez, la fragmentación debe ser **completa**, todo elemento de la relación debe estar en alguno de los fragmentos; **reconstruible**, la relación inicial debe poder reconstruirse aplicando operadores sobre los fragmentos; y **con intersección vacía**, la intersección de los fragmentos debe ser vacía (a excepción de los fragmentos).

Posteriormente, hay que **asignar** fragmentos a los esquemas locales, hay 3 tipos de asignación:

- **Sin replicación**: todo fragmento reside en un único nodo. Positivo para actualizaciones, negativo para consultas.
- **Replicación total**: todos los fragmentos residen en todos los nodos. Positivo para consultas, negativo para actualizaciones.
- **Replicación parcial**: compromiso entre actualizaciones y consultas.

	REPLICACIÓN COMPLETA	REPLICACIÓN PARCIAL	SIN REPLICACIÓN
PROCESAMIENTO DE CONSULTAS	Más fácil	Más difícil	Más difícil
CONTROL DE CONCURRENCIA	Difícil	Más difícil	Más fácil
DISPONIBILIDAD DE LOS DATOS	Muy alta	Alta	Baja

ARQUITECTURA

1. <u>Distribución</u>:

- Una BD es distribuida si está dividida en distintos componentes (integrados).
- BD distribuida != varias BD no integradas.
- Los componentes distribuidos que constituyen una BD distribuida son, a su vez **BD componentes** o locales.
- Las BD componentes tendrán un grado de autonomía local determinado.

2. <u>Autonomía</u>: tipo de control que el SGBD tiene sobre cada BD local.

- **Autonomía de diseño**: existe si los administradores de la BD pueden cambiar el esquema conceptual de sus BD independientemente de si forman parte de un sistema distribuido.
- **Autonomía de comunicación**: si se puede decidir localmente cuándo comunicarse con los otros SGBD locales.
- **Autonomía de ejecución**: si se puede ejecutar transacciones globales y locales en el orden que se quiera.
- **Autonomía de participación**: si se puede decidir cómo participar en el sistema distribuido.

3. <u>Heterogeneidad</u>:

- Distintos hardware, SO, software de comunicaciones.
- Distinto modelo de datos (relacional, jerárquico, en red, OO, etc.)
- Distintos SGBD (aunque sean del mismo modelo)

7.3. BASES DE DATOS FEDERADAS

Surgen de la unión de dos bases de datos autónomas. Proporcionan un esquema global que se obtiene de **abajo a arriba** (los esquemas locales son pre-existentes y se integran en un esquema global). No hay que fragmentar y la redundancia probablemente ya existe.

El problema de obtener un esquema global a partir de N esquemas locales se divide en dos:

- **Traducción**: cada esquema local se traduce a un modelo canónico.
- **Integración**: los esquemas locales se integran en uno solo.

ARQUITECTURA

La heterogeneidad en las B.Federadas es inherente

<u>Heterogeneidad</u>:

Además de las características de heterogeneidad de las BDD, tienen algunas más:

- Heterogeneidad semántica:
  - Sinonimia: elementos iguales con distintos nombres.
  - Homonimia: elementos distintos con el mismo nombre.
  - Otras relaciones semánticas (hiperonimia, hiponimia, agregación,etc)
  - El mismo elemento del mundo real puede ser representado como entidad o atributo.

1. Traducción: Cada uno de los esquemas locales se convierte a un modelo canónico. Se identifican entidades, relaciones y atributos, siendo posible la aparición de nuevas entidades y el cambio de nombre de algunas.
2. Integración: Los esquemas locales en modelo canónico se unifican bajo un único esquema global. Es complejo porque hay que tener en cuenta la sinonimia, herencia, uniones (algunos datos pueden estar en dos esquemas locales con distinto nombre, por ejemplo), etc

- Puede existir tanto a nivel intensional (tabla alumnos en BD1 es tabla estudiantes en BD2) como extensional (El "J. Pérez" de "alumnos.nombre") es equivalente al valor "Pérez, J" de "estudiantes.nombre").

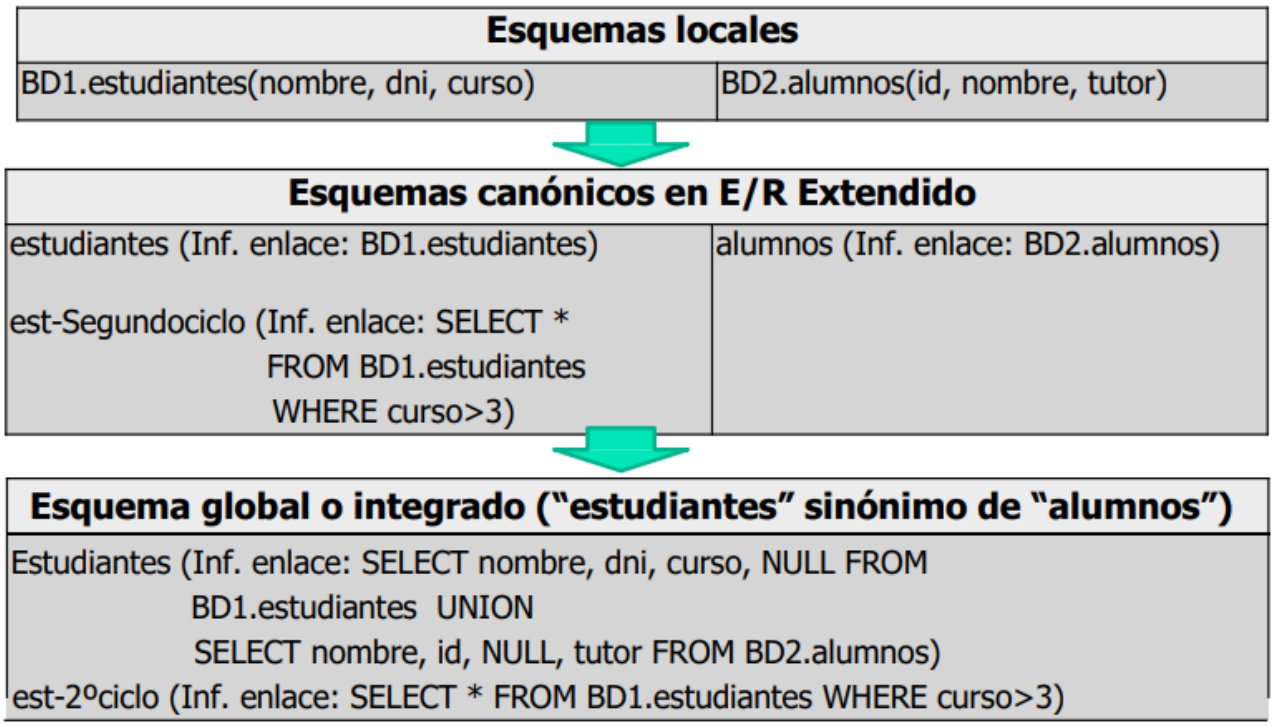
DISEÑO DE BD FEDERADAS

El diseño parte de una estructura **Bottom-Up**, en el que los pasos son:

- **Traducción**: cada esquema local se traduce a un modelo canónico (relacional).
- **Integración**: los esquemas locales se integran en uno solo.

El **modelo de datos (canónico)** utilizado para expresar el esquema global es muy importante ya que no hay que olvidar que las bases de datos locales pueden ser heterogéneas y puede que se usen modelos más ricos semánticamente que el relacional.

A la hora de realizar **consultas** sobre el esquema global, estas se deben responder sobre los **esquemas locales**. Para ello, hay que preservar la **información de enlace**, es decir, la relación entre los elementos del esquema global y los de los esquemas globales.



7.4. BASES DE DATOS INTEROPERANTES

Están formadas por BD autónomas y no proporcionan un esquema global sino lenguajes de acceso a BD. El usuario es consciente de que trabaja con varias BD.

8. MINERÍA DE DATOS

8.1. INTRODUCCIÓN

La **minería de datos** se basa en descubrir, a partir de los datos, conocimiento interesante.

Según el objetivo general la minería de datos puede ser **predictiva** o **descriptiva**.

Los **pasos** a seguir en el proceso de minería de datos son:

1. Aprender sobre el dominio de aplicación.
2. Seleccionar los datos para analizar.
3. Preparar los datos (cleaning -> fiabilidad)
4. Reducir y transformar los datos.
5. Escoger el objetivo de la minería de datos (agrupamiento, clasificación, asociación,etc)
6. Escoger un algoritmo de minería de datos apropiado (no *free lunch*)
7. Analizar los resultados
    a. Utilizar herramientas de visualización de forma adecuada.
    b. Transformación de resultados
    c. Interpretación y extracción de conclusiones
8. Explotación del conocimiento descubierto.

Algunas de las **herramientas gráficas** de minería de datos son *Weka, RapidMiner, KNIME*. Las **librerías** son *Mahout* y *MLlib*. Los **lenguajes de programación** empleados son R y Python.

8.2. REGRESIÓN

La **regresión lineal** se trata de obtener la línea que mejor se ajusta a los datos. La **regresión no lineal** se trata de obtener el polinomio que mejor se ajusta a los datos.

8.3. MINERÍA DE PATRONES Y REGLAS DE ASOCIACIÓN

Conjuntos de items:
$X = \{x_1,\dots,x_k\}$ //conjunto de elementos del antecedente
$Y = \{y_1,\dots,y_l\}$ // conjunto de elementos del consecuente
$A = \{evento \mid evento \subset X\}$ $B = \{evento \mid evento \subset Y\}$ $A \cap B = \{evento \mid evento \subset (X \cup Y)\}$

Reglas X → Y
Si ocurre X también ocurre Y. $A \cap B$ es el conjunto de eventos que cumple la regla $X \rightarrowtail Y$, es decir, que contienen el conjunto de items $X \cup Y = \{x_1,\dots,x_k, y_1,\dots,y_l\}$.

SOPORTE

Probabilidad de que una transacción tenga X∪Y.

$$\text{soporte}(X \rightarrowtail\!\!\!\!\!\rightarrow Y) = \frac{N_{X \cup Y}}{N}$$

N = Número total de instancias.
N_{XUY} = Número de instancias que contienen X e Y
Valores entre 0 y 1 (0=ningún soporte, 1=soporte total)

CONFIANZA

Probabilidad de que una transacción tenga X∩Y

$$\text{confianza}(X \rightarrowtail\!\!\!\!\!\rightarrow Y) = \frac{N_{X \cup Y}}{N_X} = \frac{soporte(X \cup Y)}{soporte(X)}$$

N_{XUY}=número de instancias que contienen X e Y
N_x=número de instancias que contienen X
N=número total de instancias.
Valores entre 0 y 1

ELEVACIÓN

Ratio entre el soporte y el producto de las probabilidades de cada conjunto por separado

$$lift(X \rightarrow Y) = \frac{N_{X \cup Y}/N}{(N_X/N)*(N_Y/N)} = \frac{soporte(X \rightarrow Y)}{(N_X/N)*(N_Y/N)} = \frac{confianza(X \rightarrow Y)}{N_Y/N}$$

Proporción del soporte observado del conjunto de items sobre el soporte teórico asumiendo independencia entre los items. La elevación indica el incremento de la probabilidad de que ocurra el consecuente de la regla si se da el antecedente.

Si es >1 → correlación positiva (si se da X es más probable que se de Y).
Si es =1 → sucesos independientes. Da igual si se da X o no para que se de Y.
Si es <1 → correlación negativa (si se da X, es menos probable que se de Y).

8.4. AGRUPAMIENTO

Un **cluster** o agrupamiento es un conjunto de entidades tales que hay cohesión (similitud dentro del cluster) y diferenciación (disimilitud entre agrupaciones diferentes). Interesa maximizar tanto la cohesión como la diferenciación.

Los clusters tienen múltiples **utilidades**:

- Describir y entender los datos.
- Paso previo a otros algoritmos.
- Detección de outliers (valores que difieren significativamente del resto)
- Descubrir grupos de clientes y lanzar campañas comerciales dirigidas a ellos.
- Agrupar viviendas similares.

8.5. CLASIFICACIÓN

La clasificación se basa en asociar cada elemento del conjunto de datos a una serie de categorías definidas previamente.

Hay distintos tipos de **variables**:

- **Respuesta** o dependiente: etiqueta cada instancia con la categoría correspondiente.
- El resto son **predictoras** o independientes.

El objetivo es explicar la variable dependiente en términos de las variables independientes.

Entre sus **utilidades** destacan:

- Clasificar el correo electrónico en spam o ham.
- Clasificar los clientes en buenos, malos o medios.
- Clasificar automáticamente una noticia en la sección del periódico adecuada.
- Etc

MATRIZ DE CONFUSIÓN

Caso binario

		Clase detectada	
		Positiva	Negativa
Clase real	Positiva	tp	fn
	Negativa	fp	tn

Caso multiclase

		Clase detectada		
		A	B	C
Clase real	A	tp_A	e_{AB}	e_{AC}
	B	e_{BA}	tp_B	e_{BC}
	C	e_{CA}	e_{CB}	tp_C

En ambos casos la **diagonal** representa los datos bien clasificados, mientras que los elementos fuera de la diagonal son errores cometidos.

MÉTRICAS TÍPICAS

La **precisión** representa los datos de una clase identificados correctamente respecto a todos los identificados como dicha clase. Valor entre 0 y 1.

Precisión = tp/(tp + fp)	$Precision_A = tp_A / (tp_A + e_{BA} + e_{CA})$

El **recall** representa los datos de una clase identificados correctamente respecto a todos lo que son realmente de dicha clase. Valor entre 0 y 1.

Recall = tp/(tp + fn)	$Precision_A = tp_A / (tp_A + e_{AB} + e_{AC})$

El **F-measure** es una medida armónica de la precisión y el recall.

F-measure=(2*precisión*recall)/(precisión+recall)

La **accuracy** es la medida de la corrección global del modelo.

Accuracy=nº de clasificaciones correctas/nº total de clasificaciones realizadas

MÉTRICAS TÍPICAS DE LA CLASIFICACIÓN BINARIA

El **FPR** (false positive rate = false alarm rate = fallout) representa cuántos resultados se detectan como positivos de forma incorrecta entre todas las muestras negativas.

$$FPR=fp/(fp+tn)$$

El **TPR** (true positive rate = sensitivity = recall) representa cuántos resultados se detectan como positivos de forma correcta de entre todas las muestras positivas.

$$TPR=tp/(tp+fn)$$

El **FNR** (false negative rate) representa cuántos resultados se detectan como negativos de forma incorrecta de entre todas las muestras positivas.

$$FNR=fn/(tp+fn)=1-TPR$$

El **TNR** (true negative rate = specificity) representa cuántos resultados se detectan como negativos de forma correcta entre todas las muestras negativas.

$$TNR=SPC=tn/(fp+tn)=1-FPR$$

EVALUACIÓN

Para evaluar hay que separar los datos en conjunto de entrenamiento y conjunto de test (o validación). Nunca hay que evaluar sobre el mismo conjunto de datos utilizados para entrenar.

La **evaluación cruzada** de k (k-fold cross-validation) vías se realiza de la siguiente manera:

- Se particiona la muestra inicial en k muestras de igual tamaño.
- 1 de las muestras se utiliza como test y las k-1 restantes para entrenar.
- Se repite k veces, de forma que cada una de las k muestras se utilizan y vez como test.
- Se combinan los resultados de las k evaluaciones (promedios)
- Si k=n(número de muestras) => leave-one out cross-validation

8.6. MINERÍA DE TEXTOS

La **minería de datos textuales** es el proceso de derivar información de "alta calidad" a partir de fuente de texto (no estructuradas o mínimamente estructuradas). Para ello se estructura la entrada (pre-procesamiento del texto) y se aplica minería de datos sobre los datos estructurados.

Las **tareas típicas** son:

- Clasificación de textos (categorización).
- Agrupación de textos (extracción automática de temas).
- Extracción de información (entidades y sus relaciones, datos de interés, correferencias)
- Análisis del sentimiento / minería de opiniones.
- Generación de resúmenes.

TÉCNICAS

En cuanto a las tareas de pre-procesamiento de textos destacan las siguientes técnicas:

- Reconocimiento de caracteres (*OCR*)
- Tokenización
- Lematización
- Stemming
- Etiquetado gramatical
- Análisis sintáctico (parsing)
- Desambiguación

Los documentos se representan de **forma vectorial**, es decir, se representan en un espacio vectorial multidimensional => bolsas de palabras. Los **términos** son las dimensiones del espacio y los **documentos** son puntos o vectores en este espacio. El valor de cada **componente** del vector se determina a partir de la frecuencia del término en el documento y su frecuencia inversa. La **similitud** entre los documentos puede calcularse midiendo el ángulo formado por sus vectores. Para mejorar el rendimiento, podrían seleccionarse únicamente las palabras más frecuentes.

9. ALMACENES DE DATOS

9.1. INTRODUCCIÓN

Dado que las organizaciones manejan enormes cantidades de datos en distintos formatos, que residen en distintas bases de datos y que están organizados utilizando distintos tipos de gestores de datos, resulta difícil acceder y utilizar todos los datos en aplicaciones de análisis.

Un **Data Warehouse** es un repositorio de datos estructurados a nivel de empresa, con datos históricos y actuales, que facilita la toma de decisiones.

Tipos de sistemas de información:

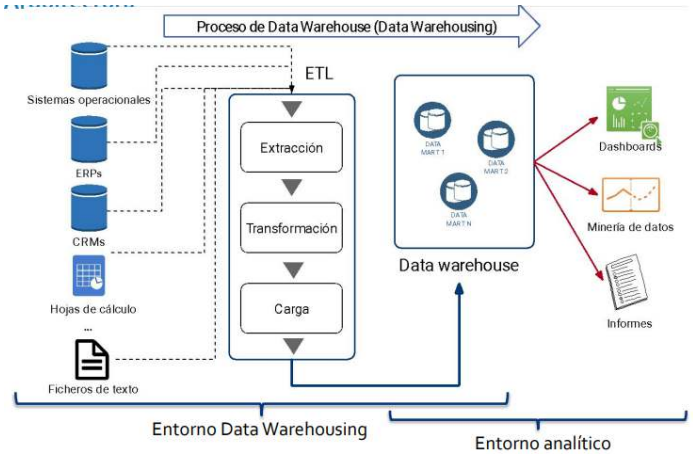
Transaccionales (OLTP)	Analíticos (OLAP)
Datos operacionales	Datos consolidados (suelen provenir de distintas BD OLTP)
Muchas transacciones (INSERT, UPDATE, DELETE)	Pocas transacciones
Datos actuales	Datos actuales e históricos
Información detallada	Información detallada y resumida (integrada) (Consultas complejas – agregaciones □ Data mining)
Los datos cambian continuamente (volátiles)	Datos con mayor estabilidad y menos cambios (no volátiles)

Las **características** de los almacenes de datos son las siguientes:

- **Orientados a un aspecto concreto**: la información se guarda en base a un tema de interés para los directivos de la entidad.
- **Integrados**: el almacén de datos suele contener todos los datos de los sistemas operacionales de la empresa. Dichos datos suelen ser consistentes.
- **No volátiles**: una vez los datos han sido incorporados al sistema (registrados) no se borran ni actualizan. Además están pensados para un horizonte de tiempo mucho mayor que los datos operacionales.

9.2. CONSTRUCCIÓN DE DATA WAREHOUSES

ARQUITECTURA



PROCESOS ETL

Primero se realiza una **extracción** de datos de fuentes de datos heterogéneas (BD relacionales con datos estructurados, semi-estructurados o no estructurados). Puede llevarse a cabo para realizar una imagen inicial o para actualizar una imagen ya existente. Es muy costoso en tiempo por lo que puede afectar al rendimiento de los sistemas de fuentes de datos.

Luego se realiza una **carga** de los datos de la transformación. Hay dos métodos:

- carga completa: imagen inicial.
- carga incremental: carga en intervalos de tiempo regulares y planificados. Se puede hacer en streaming (volúmenes pequeños de datos) o por lotes (volúmenes grandes). Se realiza un mantenimiento de históricos.

El **staging area** facilita los procesos de extracción y transformación de los datos antes de ser incluidos en el Data Warehouse

MODELOS MULTIDIMENSIONALES

Para organizar los datos de un DW se usan **cubos n-dimensionales** o **hipercubos**. En cada una de las dimensiones hay un diferente nivel de detalle.

Un **slice** (loncha) es el subconjunto de datos multidimensionales definidos por seleccionar valores específicos para cada uno de los atributos que definen las dimensiones.

Hay dos formas de **implementación** de cubos:

- **Virtual**: opción más simple. Una sola tabla con múltiples columnas que representan o bien las dimensiones que se consideran o bien los datos de interés para el análisis. Se sigue un esquema de estrella
- **Física**: Bases de datos multidimensionales. Se crea una matriz n-dimensional almacenando los valores.

La **arquitectura en estrella** se basa en una tabla central que contiene la información de los hechos que se desea analizar conectada a las diferentes tablas que representan las diferentes dimensiones.

Se pueden generar **informes** configurables. Hay dos operadores definidos sobre los informes:

- *Drill down*: detallar los resultados obtenidos añadiendo un campo.
- *Roll-up*: agregar los resultados obtenidos eliminando un campo.

Esta arquitectura tiene varios **factores de éxito**:

- Integra datos externos con los datos de producción internos y gestiona historiales.
- Considera información útil, centrada en los objetivos de la empresa.
- Emplea datos de calidad (coherentes, actualizados y documentados).
- Es flexible para garantizar escalabilidad.

Los errores más comunes a la hora de crear un DW son los siguientes:

- Incluir datos solamente porque están disponibles (pueden no ser útiles)
- Crear un esquema de BD relacionales tradicionales.
- Crear el DW pensando en la tecnología que se va a usar para su implementación.
- Creer que los DW acaban su ciclo de vida una vez son cargados los datos e instalado el sistema.