ENGINYERIA EN INFORMÀTICA

Memòria Projecte Final de Carrera

GENERACIÓ, ACTUALITZACIÓ Y CONSULTA DE BASES DE DADES RELACIONALS UTILITZANT XML (SERVEI D'EMMAGATZEMAMENT DE DADES A INTERNET)

Autor: Alejandro Chueca Barrios
Consultor: Alex Alfonso Minguillón

Juny – 2004

Títol

Generació, Actualització y Consulta de bases de dades relacionals utilitzant XML (Servei d'Emmagatzemament de dades a Internet)

Agraïments

A la meva dona, la Esther, que durant els tres anys que he estat estudiant no ha fet altra cosa que donar-me suport. Gràcies.

Resum

El que es pretén, en aquest projecte, és generar una sèrie d'eines que permetin, a una entitat externa, interactuar en una base de dades relacional sense necessitat de coneixements específics de bases de dades. Quant es diu interactuar, es vol incloure des de la creació de la base de dades, fins a la consulta de la informació passant per l'actualització d'aquesta.

Índexs

Índex de Continguts

Índex de Figures

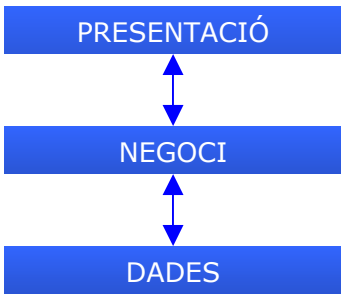
Cos de la Memòria

1.1 Introducció

1.1.1 Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC

Dintre d'aquest projecte el que es vol implementar és un servei que ofereix, dintre d'Internet, la possibilitat d'emmagatzemar dades de forma desatesa. Es a dir, que qualsevol usuari de la Xarxa pugui generar de nou una base de dades amb la informació que necessiti i, posteriorment, que pugui fer insercions, modificacions, eliminacions i consultes d'aquesta informació. No li importa quina base de dades hi ha al darrera, quines sentències SQL cal implementar, quins controls de coherència cal preservar ... Únicament s'ha de preocupar de implementar de forma correcta el protocol de comunicació amb el nou Servei d'Emmagatzemament de Dades.

Actualment, la tendència de les aplicacions informàtiques que es desenvolupen és la de diferenciar l'aplicació en diferents capes. Bàsicament, la diferenciació es fa en tres capes, una és la responsable de la visualització de les dades (Lògica de Presentació), un altra és la que s'encarrega d'implementar les diferents casuístiques que vol resoldre l'aplicació (Lògica de Negoci) i, finalment, la capa de gestió de la base de dades (Lògica de Dades). Habitualment, la Capa de Dades correspon a un Sistema de Gestió de Bases de Dades que facilita la tasca d'emmagatzemament de la informació. Oracle, SQL-Server, Informix, My-SQL... són exemples de SGBD.



Aquest projecte es basa en la intenció de facilitar la implementació de la capa de dades a les aplicacions que es desenvolupen per estar presents a Internet. Es pretén afegir un nivell més d'abstracció que permeti interactuar en bases de dades relacionals sense necessitat de gestionar-les directament. Per aconseguir-ho ha set necessari dissenyar un protocol d'intercanvi d'informació i una sèrie d'eines (interfaces) capaces de processar les diferents peticions rebudes.

Per posar un exemple, una web dissenyada en continguts totalment estàtics podria voler dinamitzar els seus continguts en, per exemple, una gestió de notícies. El dissenyador de la web hauria de plantejar-se el disposar d'una base de dades on emmagatzemar aquesta informació amb tota la problemàtica que això pot suposar, com establir els accessos amb la base de dades (ODBC, JDBC, ADO...) o aconseguir un hosting per la web que inclogués la base de dades que volem utilitzar. A més, molt probablement, caldrà que el desenvolupador tingui coneixements de SQL (Structured Query Language). Si el dissenyador disposés d'una eina que li facilités la creació d'una base de dades i la seva gestió, es solucionaria d'una forma senzilla la seva problemàtica. També aconseguiria una total independència entre la base de dades que s'està utilitzant i el dispositiu que sol·licita o genera la informació.

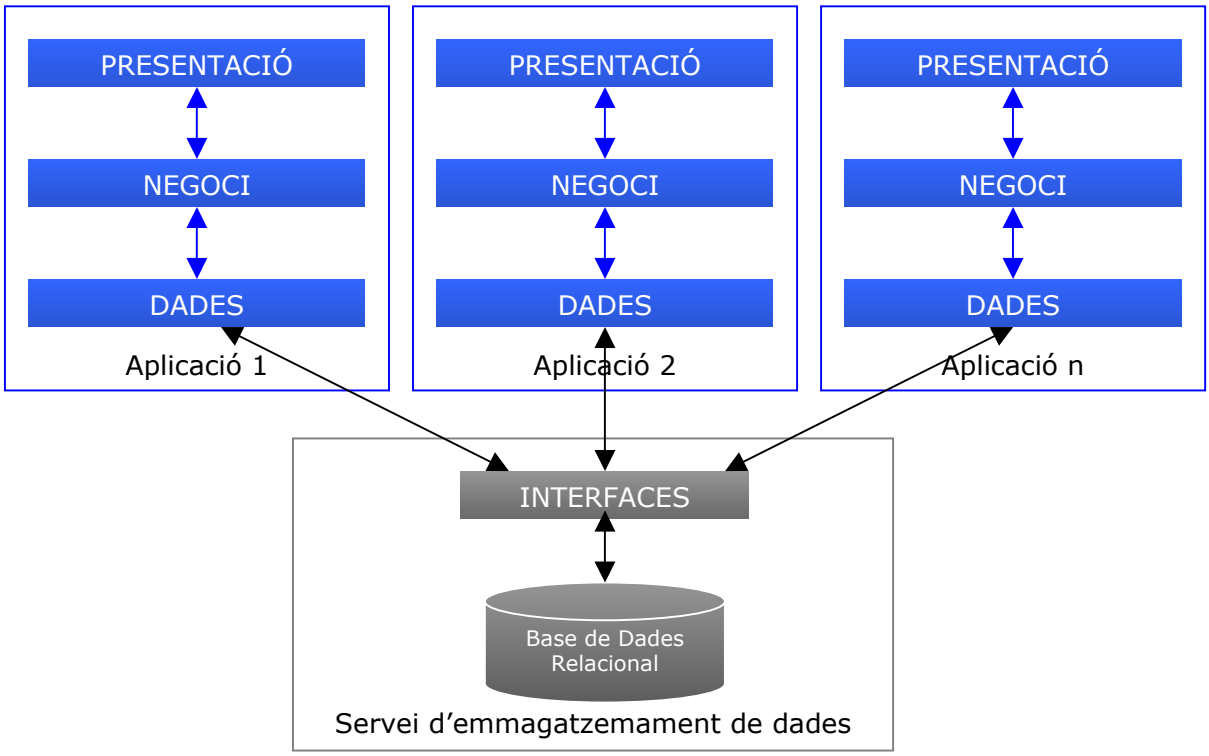


Figura 1 *Estructura de n capes aplicacions*

Per implementar el protocol d'intercanvi d'informació entre les capes de dades de les aplicacions i el Servei d'Emmagatzemament de Dades, s'ha triat un llenguatge com és XML (en un altre apartat d'aquesta memòria es detallen les característiques d'aquest llenguatge i es justifica la seva utilització com a eina de comunicació a aquest projecte). El Servei d'Emmagatzemament s'encarrega de rebre les sol·licituds d'informació de les aplicacions client expressades en arxius XML i retorna les respostes també en arxius XML.

Continuant amb l'exemple anterior, des d'una pàgina web, des d'un telèfon mòbil, des d'un PDA... es podrien enviar i rebre arxius XML amb actualitzacions i consultes de dades. La única diferència seria la representació de la informació davant l'usuari final ja que es faria en HTML, WML... o amb el llenguatge que fos necessari (aquesta característica és molt fàcil d'implementar amb un dels components del sistema XML com és el XSLT).

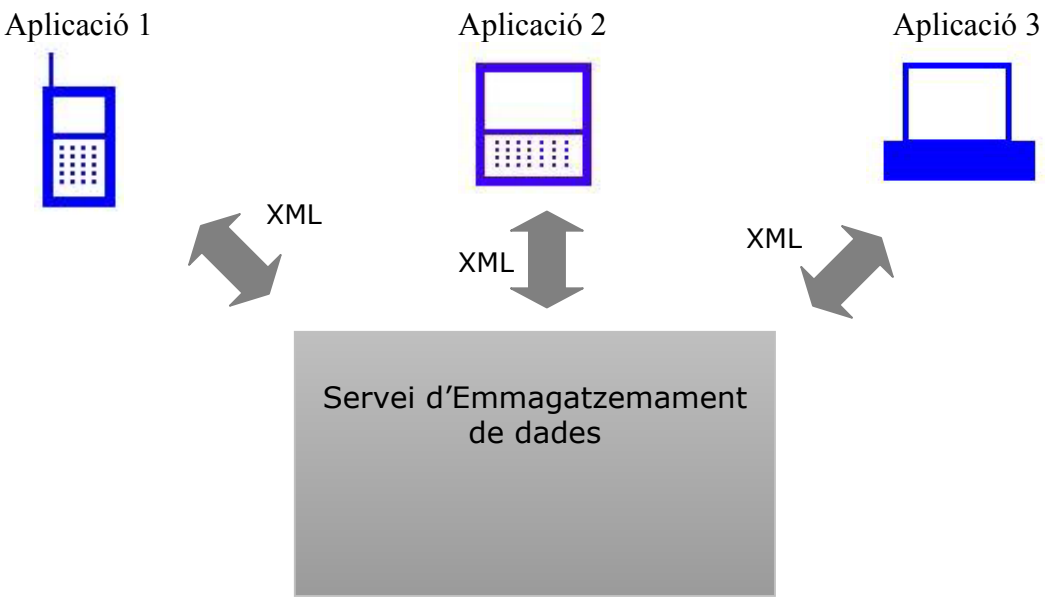


Figura 2 *Relacions Servei d'Emmagatzemament de dades*

A la figura que es mostra a continuació es pot veure, de forma detallada, un esquema de la estructura interna del Sistema d'emmagatzemament de dades i de la informació que rep i genera.

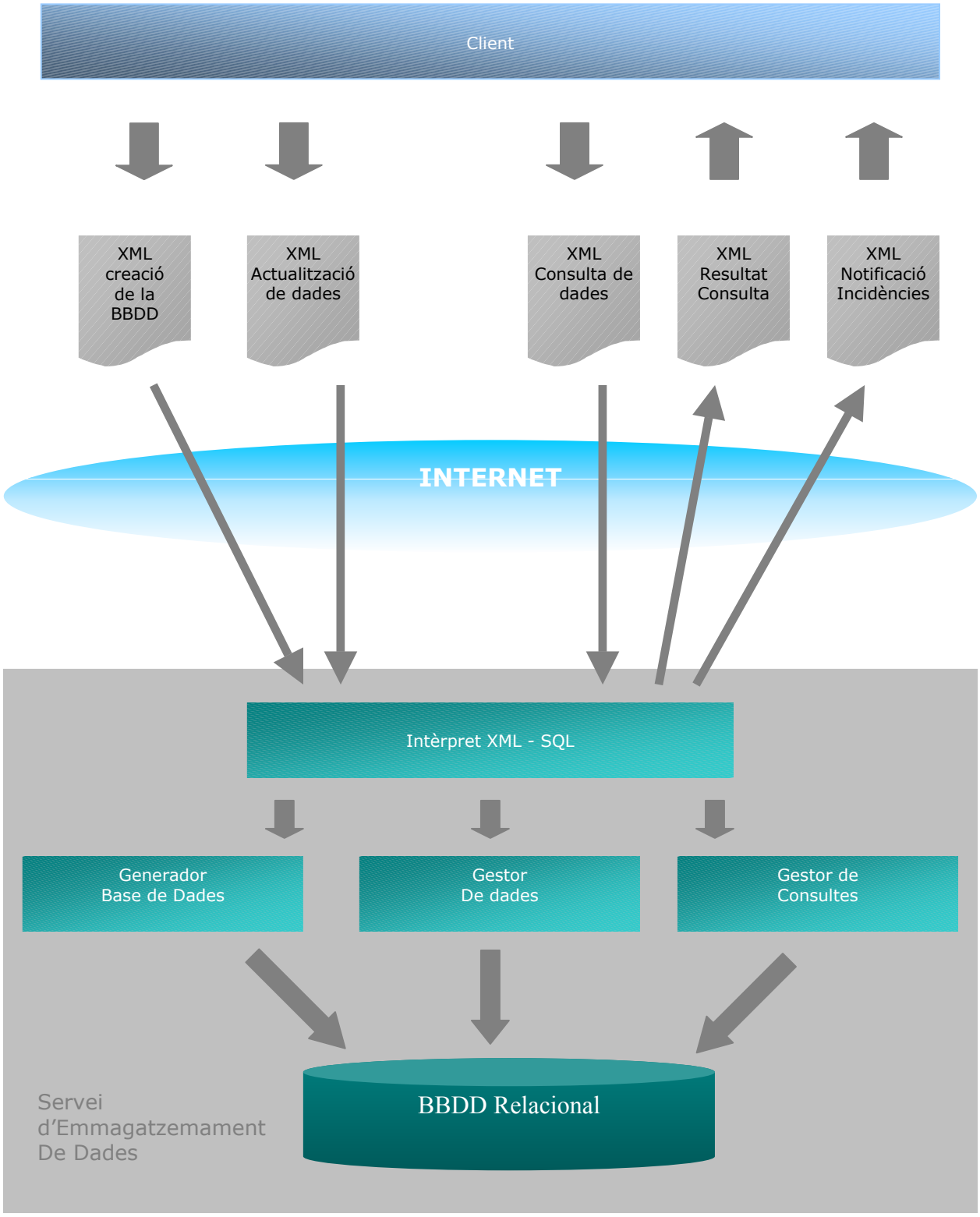


Figura 3 *Esquema del servei d'emmagatzemament de dades*

1.1.2 Objectius

Els objectius d'aquest projecte els podem resumir en quatre apartats:

- Estudi de les capacitats de generació d'esquemes de bases de dades relacionals a partir de diagrames basats en XML.
- Definir la transformació entre els elements del diagrama i una base de dades relacional.
- Construcció d'una aplicació capaç d'interpretar consultes fetes en estructura XML sobre una base de dades relacional i de construir fitxers de resposta XML amb la informació obtinguda.
- Construcció d'una aplicació capaç d'actualitzar les dades d'una base de dades relacional a partir de fitxers d'origen XML.

1.1.3 Enfocament i mètode seguit

Aquest projecte pretén aprofundir en l'intercanvi d'informació des de diferents tipus d'aplicacions client i una base de dades relacional. L'enfocament del projecte s'ha concretat en un llenguatge clarament en expansió com és XML i en una base de dades relacional com és Microsoft SQL – SERVER. Des d'aquest projecte fem una defensa de les avantatges que comporta la utilització de XML i, en quant a la base de dades, les conclusions del projecte serien bastant similars si se'n hagués utilitzat qualsevol altra com ORACLE, INFORMIX, My-SQL... No és objectiu d'aquest projecte l'anàlisi dels Sistemes Gestors de Bases de Dades. Segurament la implementació d'un Servei d'emmagatzemament de dades a Internet requeriria la implementació de moltes més tasques a part dels interfaces que s'han desenvolupat però, aquí, s'ha mostrat un punt de partida. En altres punts d'aquesta memòria s'indiquen possibles vies d'ampliació.

La metodologia utilitzada a aquest projecte està seguint un cicle de vida dividit en cinc fases:

Aprovació: dintre d'aquesta fase el consultor va proposar el PFC al Cap del departament de Bases de Dades de la UOC. Va ser validat i acceptat i es va incloure a la oferta de PFC.

Definició i Planificació: com alumne, vaig escollir aquest projecte i vaig elaborar un Pla de Treball on és definia el projecte (incorporant variacions de forma justificada al projecte original), s'especificaven els objectius i s'establia una planificació temporal. Aquesta planificació es va presentar al consultor de l'assignatura per a que la validés.

Execució: Una vegada aprovada la planificació vaig iniciar el desenvolupament del projecte. Aquest desenvolupament ha anat sent controlat per part del consultor mitjançant diferents lliuraments parcials (PAC) segons les fites establertes.

Tancament: En aquesta fase es troba actualment el projecte. S'ha preparat la memòria final del projecte indicant si s'han assolit el objectius indicats a l'inici. Finalment, serà el Comitè Avaluador del PFC qui determinarà el grau d'acompliment d'aquestos objectius.

1.1.4 Planificació del projecte

La planificació del projecte es va fer tenint en compte les dates dels diferents lliuraments parcials (PAC) i els diferents interfaces a generar. Es a dir, es va subdividir la feina a realitzar en vàries sub-tasques. Posteriorment, es va fer una valoració del volum de treball que significava cadascuna de les tasques. A continuació es pot veure una taula resumen de les principals tasques i un diagrama de GANTT on es pot veure la planificació temporal i les fites principals.

Tasca	Duració (en jornades)	Data Límit
Pla de treball	9 jornades	12/03/04
Interface de Creació de Taules	23 jornades	13/04/04
Interface de Gestió d'Actualitzacions	13 jornades	26/04/04
Interface de Gestió de Consultes	11 jornades	17/05/04
Elaboració de la memòria i presentació	24 jornades	21/06/04

Figura 4 *Taula de Tasques del Projecte*

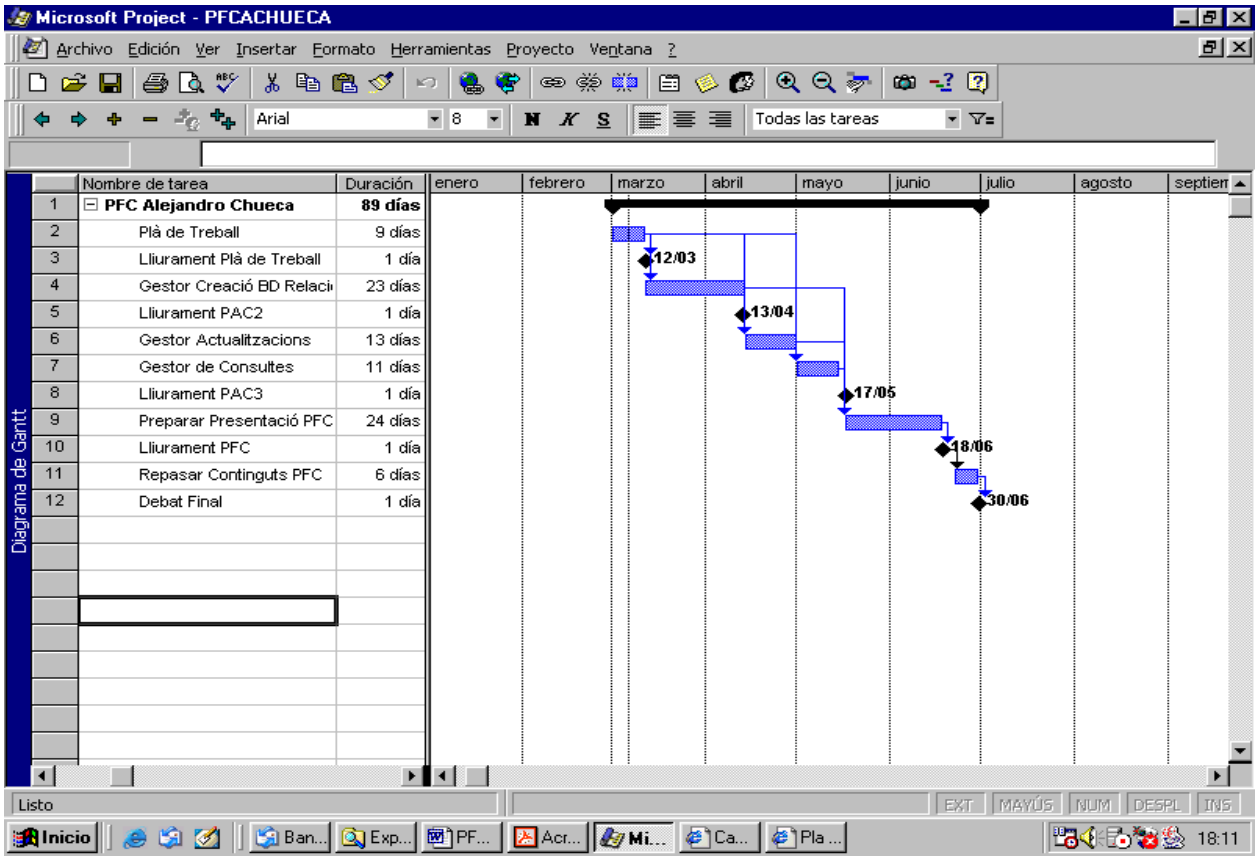


Figura 5 *Diagrama de Gantt del Projecte*

1.1.5 Productes obtinguts

Com a productes resultants de portar a terme aquest projecte i que podrien ser utilitzats com a base per a posteriors ampliacions d'aquest, podem anomenar els interfaces (processos de gestió dels fitxers XML i de control d'accés a la base de dades relacional) i els fitxers XML d'exemple.

Els interfaces desenvolupats són els següents:

- Interface de Creació de Taules: procés que interpreta els arxius XML rebuts i generà la taula corresponent a la base de dades.
- Interface de Gestió d'Altes: procés que interpreta els arxius XML rebuts, realitza els controls de validesa necessaris i insereix a la taula corresponent un nou registre.
- Interface de Gestió de Baixes: procés que interpreta els arxius XML rebuts i elimina el registre o registres corresponents.
- Interface de Gestió de Modificacions: procés que interpreta els arxius XML rebuts, realitza els controls de validesa necessaris i modifica els valors indicats al registre o registres corresponents.
- Interface de Consulta d'Informació: procés que interpreta els arxius XML rebuts, i retorna un arxiu XML amb la informació sol·licitada.

Els arxius XML generats són els següents:

- XML creació base de dades
- XML alta dades
- XML baixa dades
- XML modificació dades
- XML consulta dades
- XML resultat consulta
- XML log incidències

1.2 Per què XML?

1.2.1 Què és XML?

L'(Extensible Markup Language) és un metallenguatge que ens permet definir llenguatges de marcat adequats per usos determinats. És un estàndard internacionalment reconegut que no pertany a cap empresa i la seva utilització és totalment lliure. Va ser creat al 1996 a partir d'un altre llenguatge existent, el SGML (Standard Generalised Mark-up Language) i, al febrer de 1998, va ser adoptat com a estàndard pel World Wide Web Consortium (W3C, http://www.w3c.org).

Els documents XML poden ser estructurats per identificar cada peça important d'informació (així com les relacions entre aquestes peces) i permeten que s'escrigui codi que processi aquestos documents sense intervenció humana.

A continuació es mostra un exemple d'un arxiu XML que es podria utilitzar per emmagatzemar la informació corresponent a un alumne de la UOC:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Estat SYSTEM "DadesAlumne.dtd">
<DadesAlumne>
    <dni>18256325K</dni>
    <Nom>Alejandro Chueca</Nom>
    <Adreça>Carrer San Pau 3</Adreça >
    <Assignatura>
            <Nom>Arquitectura Sistemes</Nom>
            <Nota>B</Nota>
    </Assignatura>
    <Assignatura>
            <Nom>Sistemes Operatius</Nom>
            <Nota>C+</Nota>
    </Assignatura>
</DadesAlumne>
```

Figura 6 *Arxiu XML d'exemple*

Aquest és un document ben format en XML, es a dir, és tracta d'un document que segueix les normes establertes de creació d'arxius XML. Per exemple, que tots els elements tenen una etiqueta d'obertura i un altra de tancament, que les etiquetes és troben correctament estructurades ...

Però, per gestionar correctament arxius XML cal que, a més de que l'arxiu estigui ben format, aquest sigui vàlid. És pot considerar que un arxiu XML és vàlid quant s'ajusta a les restriccions especificades al arxiu DTD (Document Type Definition) o esquema (XML Schema) amb el qual se l'ha relacionat. Els DTD i esquemes són arxius on s'especifiquen les regles que volem que segueixen les dades del arxiu XML.

Una opció per a donar format a un arxiu XML són els arxius XSL (Extensible Stylesheet Language). Si per exemple s'intenta obrir un arxiu XML amb un navegador com pot ser Internet Explorer el que veuríem no seria massa diferent del que es pot veure a la figura 1. Però, en canvi, si s'utilitza per exemple un arxiu XSLT on es defineix un format de conversió per exemple a HTML, al navegador es podria veure la mateixa informació però amb un format visual totalment diferent.

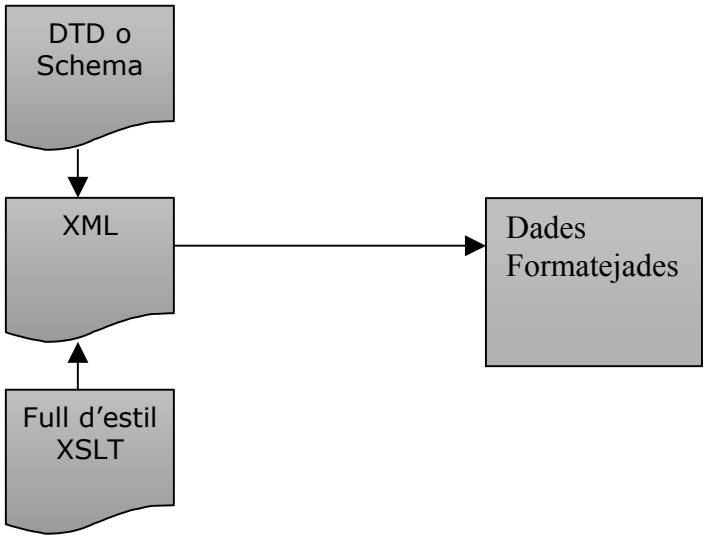


Figura 7 *Components sistema XML*

1.2.2 Avantatges

Algunes de les principals avantatges de la utilització de XML són les següents:

- Separa radicalment el que és la informació que conté de la seva representació gràfica o format (a diferència d'altres llenguatges com per exemple HTML). El mateix document pot tenir diferents formes de presentació.

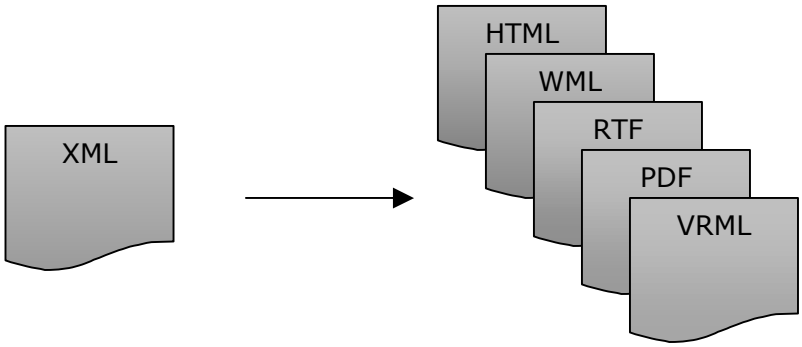


Figura 8 *Formes de presentació arxius XML*

- La seva senzillesa permet facilitar el seu processament tant per qualsevol persona com per programari.
- Les estrictes regles per la formació d'un document XML faciliten la seva anàlisi sintàctica.

Avui en dia les organitzacions utilitzen diferents formes de representar la seva informació, fins i tot dintre d'una mateixa organització. La comunicació entre aquestos, per dir-ho d'alguna forma, grups d'informació, fa que es generin grans volums de feina. És aquí on entra l'(Extensible Markup Language). XML es una eina, flexible e independent, que permet establir una comunicació entre diferents formats de dades interns de les organitzacions. Això implica reduccions de costos i estandardització de l'intercanvi d'informació.

1.3 Implementació de les eines d'intercanvi d'informació (interfaces)

1.3.1 Introducció

Un cop ja comentats quins són els objectius d'aquest projecte i quines són les característiques del llenguatge XML, passem a detallar com s'han implementat les eines que interpreten i generen els fitxers XML (interfaces de traspàs). En principi, calia triar una base de dades relacional on fer la gestió de les dades i un llenguatge amb el qual implementar els interfaces. Em vaig decidir per SQL-Server 2000 com a base de dades relacional i pels Procediments Emmagatzemats d'SQL-Server com a eina de programació. Els procediments emmagatzemats utilitzen bàsicament llenguatge SQL i s'executen dintre de la mateixa base de dades, per la qual cosa són molt ràpids. Vaig triar aquesta base de dades per que vaig llegir alguns comentaris sobre la incorporació d'eines de gestió XML a la versió 2000 i em vaig decidir a provar-les. De la mateixa forma, s'hagués pogut utilitzar altres base de dades relacionals i/o altres llenguatges de programació.

1.3.2 Programari i maquinari utilitzats

El maquinari utilitzat és un Ordinador Personal amb un processador PIV i 1 GB de RAM (punt estàndard de treball UOC).

En quant a programari he utilitzat bàsicament l'SQL-Server versió 2000. A calgut que m'instal·les a una màquina el Sistema Operatiu Windows 2000 Server ja que així ho requeria la base de dades. A part d'això he utilitzat també llenguatge .asp i el servidor de pàgines web del IIS (Internet Information Server) per fer proves d'intercanvi d'informació entre una pàgina web i els interfaces implementats.

1.3.3 Descripció de les taules internes del interfaces

Dintre de la base de dades relacional, com a taules necessàries pel funcionament dels interfaces, s'ha creat una estructura de taules relacional mínima per garantir el funcionament. Aquesta estructura està formada per tres taules, una on estan definits els clients (TMACLI) i es detalla, entre d'altra informació, l'idioma que volen utilitzar, un altra on es defineixen els missatges d'incidència (TMADIN) en els diferents idiomes i segons el seu nivell de gravetat (informació, error lleu o error greu) i, per últim, la taula de log on s'emmagatzemen totes les operacions fetes sobre la base de dades (TINLOG).

Dintre d'aquest projecte no s'ha implementat cap procés d'actualització sobre aquestes taules ja que no es considerava dintre dels seus objectius. Simplement, s'han actualitzat les dades directament sobre les taules de la base de dades quant ha set necessari.

A continuació es mostren exemples del contingut d'aquestes taules i una breu explicació de la utilitat de cadascuna:

Figura 9 *Taula de Definició de Clients(TMACLI)*

Dades:

Client	Idioma	Nom	Adreça	Correu
1	1	ALEJANDRO	SAN PAU	alejandro@C1.com
2	2	MAR	SAN PABLO	mar@C2.com

Utilitat:

Prèviament a qualsevol utilització dels interfaces per part d'una entitat externa cal que aquesta hagi set inclosa a la taula de clients. D'aquesta forma es controla l'accés a les dades i s'obtenen altres aspectes propis de cada client, com pot ser el idioma amb el qual vol rebre els missatges d'error o l'adreça electrònica de contacte.

Figura 10 *Taula de Definició d'Incidències (TMADIN)*

Dades:

Incidència	Idioma	Descripció	Tipus
1	1	No s'ha pogut generar la taula	Error Greu
1	2	No se ha podido generar la tabla	Error Greu
2	1	El NIF es incorrecte	Error Lleu
2	2	El NIF es incorrecto	Error Lleu
3	1	Taula Generada Correctament	Informació
3	2	Tabla Generada Correctamente	Informació
4	1	El Client no existeix	Error Greu
4	2	El Cliente no existe	Error Greu
5	1	Error SQL-SERVER al generar la taula	Error Greu
5	2	Error SQL-SERVER al generar la tabla	Error Greu
6	1	No s'ha indicat un camp requerit	Error Greu
6	2	No se ha indicado un campo requerido	Error Greu
7	1	El valor del camp no es numèric	Error Greu
7	2	El valor del campo no es numérico	Error Greu
8	1	El valor del camp no es una data correcta	Error Greu
8	2	El valor del campo no es una fecha correcta	Error Greu
9	1	Error SQL-SERVER al inserir un nou registre	Error Greu
9	2	Error SQL-SERVER al insertar un nuevo registro	Error Greu
10	1	El registre ja existeix a la taula (Clau duplicada)	Error Greu
10	2	El registro ya existe en la tabla (Clave duplicada)	Error Greu
11	1	Taula Actualitzada Correctament	Informació
11	2	Tabla Actualizada Correctamente	Informació
12	1	El valor del camp no es correcte	Error Greu
12	2	El valor del campo no es correcto	Error Greu
13	1	Consulta de Dades Correcta	Informació
13	2	Consulta de Datos Correcta	Informació
14	1	Consulta d'Usuari	Informació
14	2	Consulta de Usuario	Informació

Utilitat:

Tots el missatges d'error que poden generar els interfaces es defineixen a aquesta taula. Existeixen tres tipus d'error (Greu, Lleu e Informació) i s'especifiquen varies descripcions del error en funció del idioma.

Figura 11 *Taula de Registre (TINLOG)*

Dades:

Id	Client	Procediment Emmagatzemat	Descripció	Tipus Incidència	Data/Hora
4	1	Pr_Crear_Taula	Taula Generada Correctament	Informació	12/04/04 15:30:03
5	1	Pr_Crear_Taula	Taula Generada Correctament	Informació	12/04/04 15:33:20
26	11	Pr_Crear_Taula	El Client no existeix	Error Greu	18/04/04 11:02:02
34	1	Pr_Crear_Taula	Error SQL-SERVER al generar la taula	Error Greu	27/04/04 20:13:42
56	1	Pr_Gestio_Modificacions	No s'ha indicat un camp requerit	Error Greu	09/05/04 10:39:38
70	2	Pr_Gestio_Modificacions	El valor del campo no es numérico	Error Greu	09/05/04 11:34:05
86	1	Pr_Gestio_Modificacions	Error SQL-SERVER al inserir un nou registre	Error Greu	09/05/04 10:39:38
88	1	Pr_Gestio_Modificacions	El registre ja existeix a la taula (Clau duplicada)	Error Greu	14/05/04 12:36:53
94	2	Pr_Gestio_Modificacions	Taula Actualitzada Correctamente	Informació	14/05/04 13:01:19
132	1	Pr_Consultar_Dades	Consulta de Dades Correcta	Informació	16/05/04 10:10:47
191	1	Pr_Gestio_Modificacions	El NIF es incorrecte	Error Lleu	17/05/04 11:09:53

Utilitat:

Tots els accessos que es fan a la base de dades utilitzant el Sistema d'Emmagatzemament de dades s'actualitzen a aquesta taula (ja siguin erronis o no). Es una forma de monitoritzar l'activitat de cada client.

1.3.4 Interface de Creació de Taules

Introducció:

Aquest procés s'ha implementat mitjançant procediments emmagatzemats de SQL-Server. Bàsicament el que fa es rebre la definició d'una base de dades en un document XML, l'analitza i la intenta generar a SQL-Server. Posteriorment genera un log d'incidències en format XML. Per simplificar la implementació del procés i per facilitar la traducció de l'interface a altres SGBD, únicament s'han tingut en compte tres possibles tipus de dades: text, data i numèric. L'interface també gestiona la creació dels camps clau de la nova taula ja que a l'arxiu XML passat s'incorpora una marca on s'indica si un camp és clau o no.

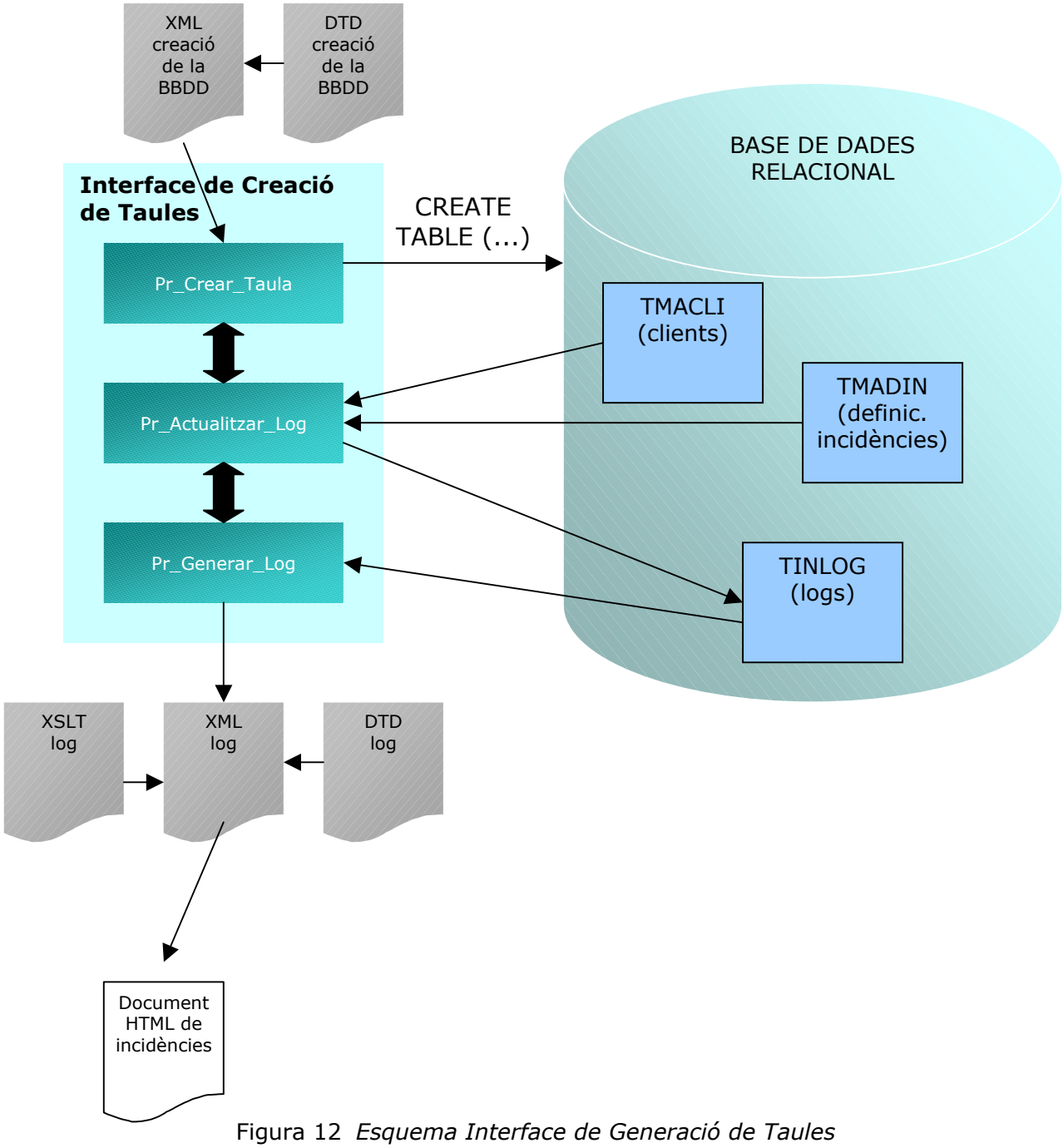


Figura 12 *Esquema Interface de Generació de Taules*

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Crear_Taula:

Descripció: És l'encarregat de llançar les instruccions SQL necessàries per a crear la nova taula a la base de dades. Obté la informació necessària d'un fitxer XML i crida al procediment Pr_Actualitzar_Log per generar un log d'incidències.

Paràmetres d'entrada: Document XML i Codi de Client.

Paràmetres de sortida: Identificador Log generat.

Crides a altres procediments:
sp_xml_preparedocument
sp_xml_removedocument
Pr_Actualitzar_Log

sp_xml_preparedocument:

Descripció: És un procediment emmagatzemat pre-definit al SQL-SERVER per tractar documents XML. Literalment als Llibres en Pantalla de SQL – Server se'l defineix com:
Lee el texto de Lenguaje de Marcado Extensible (XML) proporcionado como entrada y, a continuación, analiza el texto con el analizador MSXML (Msxml2.dll) y proporciona el documento analizado en un estado apto para su uso. Este documento analizado es una representación en árbol de varios nodos (elementos, atributos, texto, comentarios, etc.) del documento XML.

***sp_xml_preparedocument** devuelve un controlador que se puede utilizar para obtener acceso a la representación interna que se acaba de crear del documento XML. Este controlador es válido mientras dura la conexión con Microsoft® SQL Server™ 2000, hasta que se restablece la conexión o hasta que se invalida el controlador mediante **sp_xml_removedocument**.*

sp_xml_removedocument:

Descripció: És un procediment emmagatzemat pre-definit al SQL-SERVER per tractar documents XML. Literalment als Llibres en Pantalla SQL – Server se'l defineix com:

Quita la representación interna del documento XML especificada por el controlador del documento y lo convierte en no válido.

Descripció dels documents XML utilitzats:

XML Creació Taula: S'utilitza per definir la estructura de nova taula que s'ha de crear a la Base de Dades relacional.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DefTaula>
	<NomTaula>PERSONAL</NomTaula>
	<Camp>
		<Nom>Id</Nom>
		<Tipus>Numèric</Tipus>
		<Clau>1</Clau>
	</Camp>
	<Camp>
		<Nom>Nom</Nom>
		<Tipus>Text</Tipus>
		<Clau>0</Clau>
	</Camp>
	<Camp>
		<Nom>NIF</Nom>
		<Tipus>Text</Tipus>
		<Clau>0</Clau>
	</Camp>
	<Camp>
		<Nom>Adreça</Nom>
		<Tipus>Text</Tipus>
		<Clau>0</Clau>
	</Camp>
	<Camp>
		<Nom>Data_Naixement</Nom>
		<Tipus>Data</Tipus>
		<Clau>0</Clau>
	</Camp>
</DefTaula>
```

Com es pot veure l'arxiu XML conté el nom de la taula a generar especificat al tag <NomTaula> que és, en aquest cas, *PERSONAL*. Després figuren els continguts dels camps, tag <Camp>, compostos pel nom del camp, el tipus i l'indicador de si el camp és clau o no. Al exemple anterior la clau estaria formada per un únic camp amb nom *Id* i de tipus Numèric. La estructura relacional de la taula de l'exemple anterior seria la següent:

Nom Camp	**Tipus**	**Clau**
Id	Numèric	Si
Nom	Text	No
NIF	Text	No
Adreça	Data	No
Data_Naixement	Data	No

La instrucció SQL equivalent seria la següent:

```
CREATE TABLE PERSONAL (
        Id [numeric](18,0),
        Nom [nvarchar](50),
        NIF [nvarchar](50),
        Adreça [nvarchar](50),
        Data_Naixement [datetime]
        PRIMARY KEY (Id));
```

DTD Creació Taula: s'utilitza per validar el format del XML de creació de Taula.

1.3.5 Interface de Gestió d'Altes

Introducció:

Aquest procés s'ha implementat mitjançant procediments emmagatzemats de SQL-Server. Insereix noves files a les taules de la base de dades en funció del fitxer XML passat. Verifica que les dades siguin correctes, que els valors obligatoris hi siguin i, si s'han especificat, efectua les funcions de validació.

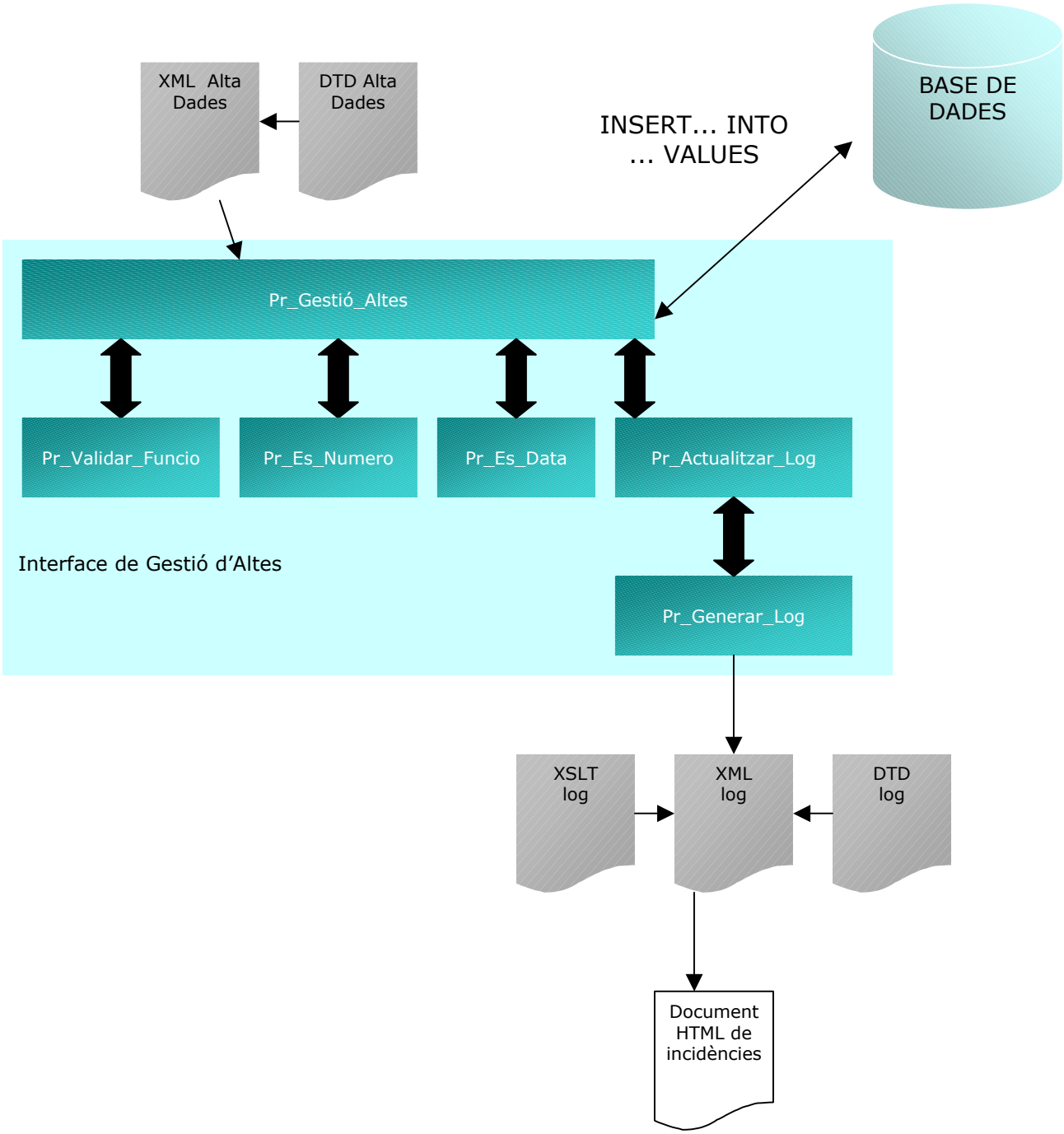


Figura 13 *Esquema Interface de Gestió d'Altes*

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Gestió_Altes:

Descripció: És l'encarregat de llançar les instruccions SQL necessàries per a incorporar nova informació a la base de dades. Obté la informació necessària d'un fitxer XML, utilitza els procediments Pr_Es_Numero i Pr_Es_Data per validar els continguts i Pr_Validar_Funcio per verificar els valors dels camps per als quals s'han definit funcions de validació. Cal comentar que per a fer una nova inserció a la base de dades no es requereixen tots els camps, únicament es comprova que hi siguin els obligatoris.

Paràmetres d'entrada: Document XML.

Paràmetres de sortida: Identificador fitxer d'incidències.

Crides a altres procediments:
sp_xml_preparedocument
sp_xml_removedocument
Pr_Validar_Funcio
Pr_Es_Numero
Pr_Es_Data
Pr_Actualitzar_Log

Pr_Validar_Funcio:

Descripció: Procediment que s'utilitza per a verificar el contingut d'un camp Segons la Funció de Validació que se li ha assignat. Aquestes funcions tenen una correspondència amb processos emmagatzemats que s'especifica a la taula TMADFU. Aquesta taula inclou les relacions entre el nom de la funció de validació, el procediment emmagatzemat i el tipus d'incidència a generar. Una vegada obtingut el procediment emmagatzemat de la taula s'executa i es retornen els resultats.

Paràmetres d'entrada: Nom Funció i valor.

Paràmetres de sortida: no n'hi han.

Crides a altres procediments:
Pr_Validar_Nif
Pr_Digit_Control

Pr_Es_Numero:

Descripció: Procediment que retorna 0 o 1 en funció de si el valor passat correspon a un valor numèric.

Paràmetres d'entrada: valor.

Paràmetres de sortida: no n'hi han.

Pr_Es_Data:

Descripció: Procediment que retorna 0 o 1 en funció de si el valor passat correspon a una data ben formada o no.

Paràmetres d'entrada: valor.

Paràmetres de sortida: no n'hi han.

sp_xml_preparedocument:

Ja comentat a un apartat anterior.

sp_xml_removedocument:

Ja comentat a un apartat anterior.

Descripció dels documents XML utilitzats

XML Alta Dades: S'utilitza per definir les dades que s'han d'actualitzar a la Base de Dades relacional.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Estat SYSTEM "AltaDades.dtd">
<AltaDades>
        <Client>1</Client>
        <NomTaula>PERSONAL</NomTaula>
        <Camp>
                <Nom>Id</Nom>
                <Valor>1</Valor>
                <Funcio></Funcio>
        </Camp>
        <Camp>
                <Nom>Nom</Nom>
                <Valor>Alejandro</Valor>
                <Funcio></Funcio>
        </Camp>
        <Camp>
                <Nom>NIF</Nom>
                <Valor>18027612M</Valor>
                <Funcio>Validar_NIF</Funcio>
        </Camp>
        <Camp>
                <Nom>Data_Naixement</Nom>
                <Valor>11/08/72</Valor>
                <Funcio></Funcio>
        </Camp>
</AltaDades>
```

A l'arxiu XML s'especifiquen, en primer lloc, el client que vol efectuar l'actualització i la taula sobre la que ho vol fer. Posteriorment hi poden haver tants tags <Camp> com camps té la taula especificada. Com ha mínim hi hauran de ser tots els camps obligatoris. Dins de cada tag <Camp> s'especifiquen el nom del camp, el seu valor i, de forma opcional, la funció de validació que s'ha d'aplicar al valor del camp. Al arxiu XML anterior es vol generar un nou registre a la taula PERSONAL amb els següents valors:

Id = 1
Nom = Alejandro
NIF = 18027612M
Data_Naixement = 11/08/72

Si considerem que la estructura de la taula PERSONAL correspon a la que s'ha comentat a un exemple anterior, es pot apreciar que no s'ha assignat cap valor al camp Adreça. Com no es tracta d'un camp obligatori el procediment crearia el nou registre a la taula amb un valor nul pel camp Adreça.

La instrucció SQL equivalent seria la següent:

INSERT INTO PERSONAL (Id, Nom, NIF, Adreça, Data_Naixement)
VALUES (1,'Alejandro','18027612M','','11/08/72');

DTD Creació Taula: s'utilitza per validar el format del XML de creació de Taula.

1.3.6 Interface de Gestió de Baixes

Introducció:

Aquest procés s'ha implementat mitjançant procediments emmagatzemats de SQL-Server. Elimina les files seleccionades d'una taula de la base de dades en funció del fitxer XML passat. Verifica que les dades de selecció passades siguin correctes.

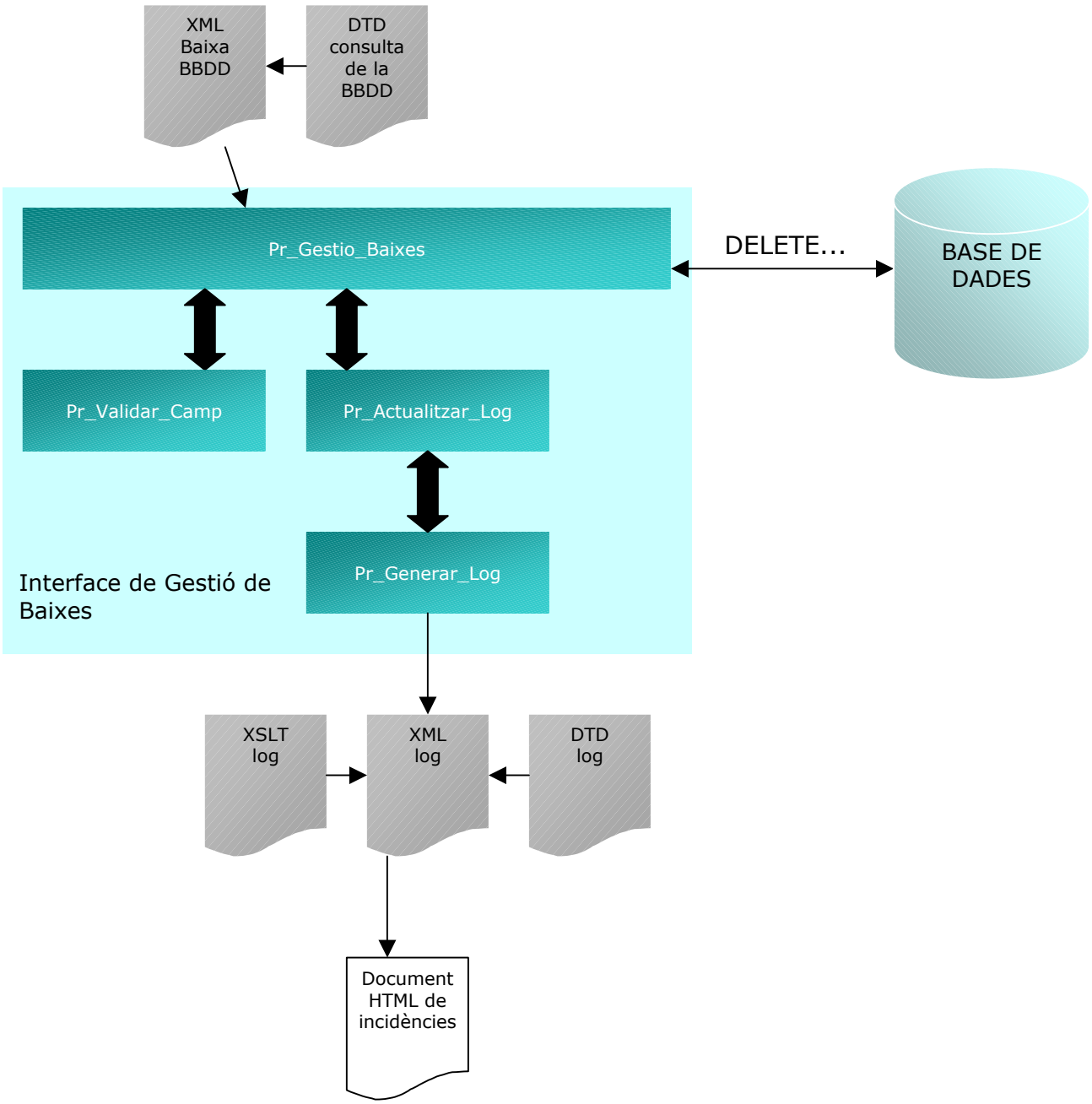


Figura 14 *Esquema Interface de Gestió de Baixes*

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Gestió_Baixes:

Descripció: És l'encarregat de llançar les instruccions SQL necessàries per a eliminar la informació sol·licitada. Obté la informació necessària d'un fitxer XML, utilitza el procediment Pr_Validar_Camp per verificar la condició de selecció i crida al procediment Pr_Actualitzar_Log per generar un log d'incidències.

Paràmetres d'entrada: Document XML.

Paràmetres de sortida: Identificador consulta generada.

Crides a altres procediments:
sp_xml_preparedocument
sp_xml_removedocument
Pr_Validar_Camp
Pr_Actualitzar_Log

Pr_Validar_Camp:

Ja s'ha comentat a un apartar anterior.

sp_xml_preparedocument:

Ja s'ha comentat a un apartar anterior.

sp_xml_removedocument:

Ja s'ha comentat a un apartar anterior.

Descripció dels documents XML utilitzats

XML Baixa Dades: S'utilitza per definir les dades que s'han d'eliminar de la Base de Dades relacional.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Estat SYSTEM "BaixaDades.dtd">
<BaixaDades>
        <Client>1</Client>
        <NomTaula>PERSONAL</NomTaula>
        <Camp>
                <Nom>Id</Nom>
                <Valor>1</Valor>
        </Camp>
</BaixaDades>
```

A l'arxiu XML s'especifiquen, en primer lloc, el client que vol efectuar l'actualització i la taula sobre la que ho vol fer. Posteriorment hi poden haver tants tags <Camp> com condicions es volen aplicar per fer l'eliminació. Dintre de cada tag <Camp> s'especifica el nom del camp i el valor que s'ha d'utilitzar per localitzar el registre a eliminar. A l'arxiu anterior es vol eliminar el registre amb Id igual a 1 de la taula PERSONAL.

La instrucció SQL equivalent seria la següent:

DELETE PERSONAL WHERE Id=1

1.3.7 Interface de Gestió de Modificacions

Introducció:

Aquest procés s'ha implementat mitjançant procediments emmagatzemats de SQL-Server. Modifica files existents a les taules de la base de dades en funció del fitxer XML passat. Verifica que les dades siguin correctes, que els valors obligatoris hi siguin i, si s'han especificat, efectua les funcions de validació.

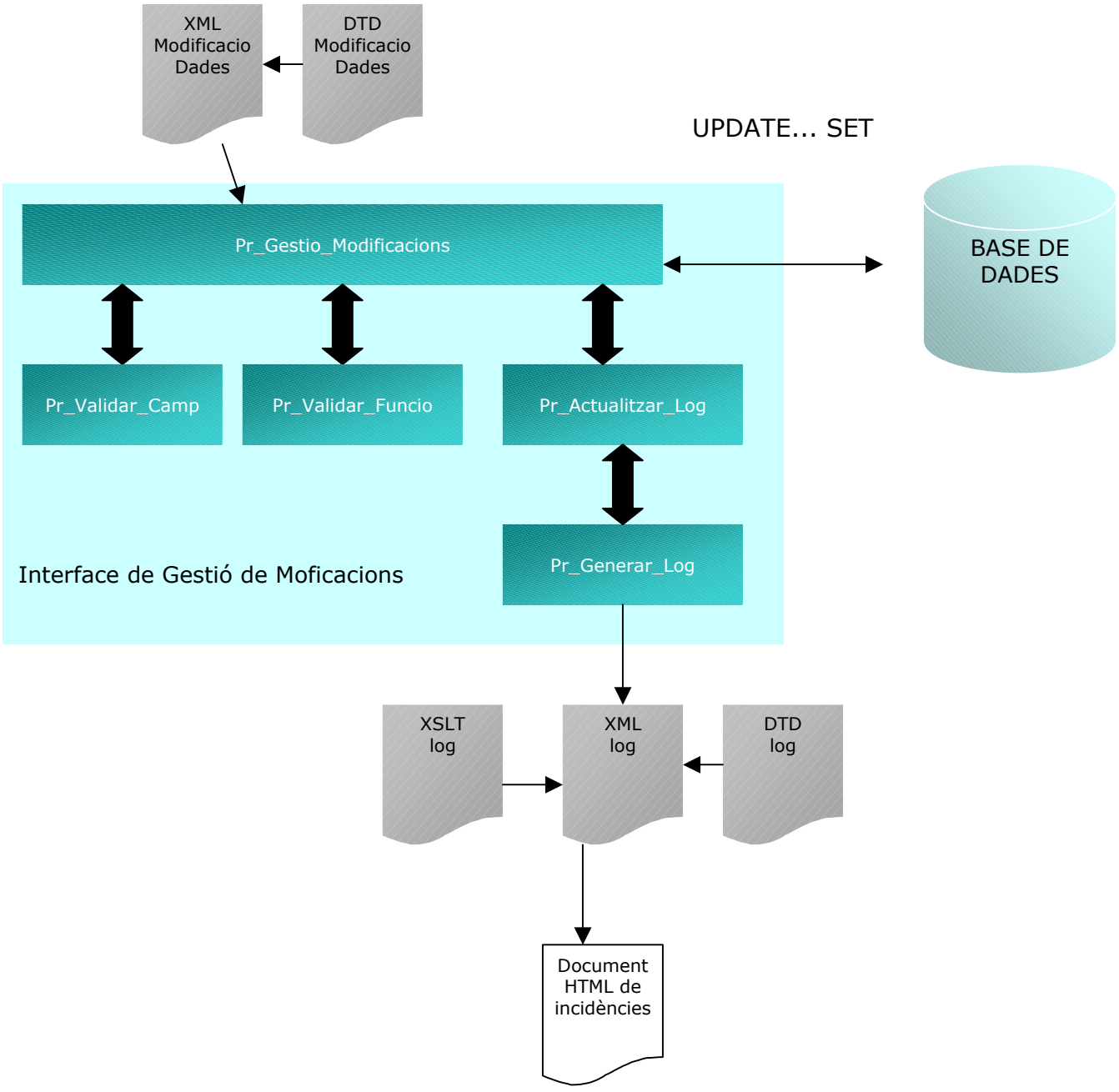


Figura 15 *Esquema Interface de Gestió de Modificacions*

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Gestió_Modificacions:

Descripció: És l'encarregat de llançar les instruccions SQL necessàries per a modificar informació existent a la base de dades. Obté la informació necessària d'un fitxer XML, utilitza els procediments Pr_Validar_Camp per validar els continguts i Pr_Validar_Funcio per verificar els valors dels camps per als quals s'han definit funcions de validació.

Paràmetres d'entrada: Document XML.

Paràmetres de sortida: Identificador fitxer d'incidències.

Crides a altres procediments:
sp_xml_preparedocument
sp_xml_removedocument
Pr_Validar_Funcio
Pr_Validar_Camp
Pr_Actualitzar_Log

Pr_Validar_Funcio:

Ja s'ha comentat a un apartat anterior.

Pr_Validar_Camp:

Ja s'ha comentat a un apartat anterior.

sp_xml_preparedocument:

Ja s'ha comentat a un apartat anterior.

sp_xml_removedocument:

Ja s'ha comentat a un apartat anterior.

Descripció dels documents XML utilitzats

XML Modificació Dades: S'utilitza per definir les dades que s'han de modificar i amb quins valors de la Base de Dades relacional.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Estat SYSTEM "ModDades.dtd">
<ModDades>
        <Client>1</Client>
        <NomTaula>PERSONAL</NomTaula>
        <CampCondicio>
                <Nom>Id</Nom>
                <Valor>1</Valor>
                <Operador>=</Operador>
                <OperadorLogic>AND</OperadorLogic>
        </CampCondicio>
        <CampCondicio>
                <Nom>Nom</Nom>
                <Valor>Alejandro</Valor>
                <Operador>=</Operador>
                <OperadorLogic></OperadorLogic>
        </CampCondicio>
        <CampMod>
                <Nom>Nom</Nom>
                <Valor>Agustin</Valor>
                <Funcio></Funcio>
        </CampMod>
</ModDades>
```

A l'arxiu XML s'especifiquen, en primer lloc, el client que vol efectuar l'actualització i la taula sobre la que ho vol fer. Posteriorment hi han dos tipus de tags, els tags <CampCondicio> on s'especifiquen els criteris de selecció dels registres i els tags <CampMod> on s'indiquen els nous valors a assignar als camps. Es interessant comentar que dintre del tag <CampCondicio> s'especifica per una banda l'operador de comparació, tag <Operador>, i per un altra l'operador lògic per enllaçar-lo amb la següent condició, tag <OperadorLogic>.

Dins de cada tag <CampMod> s'especifiquen el nom del camp, el seu valor i, de forma opcional, la funció de validació que s'ha d'aplicar al valor del camp. Si traduïm ha llenguatge SQL el contingut del camp anterior ens quedaria el següent:

UPDATE PERSONAL SET NOM='Agustin' WHERE ID=1 AND NOM='Alejandro'

1.3.8 Interface de Consulta d'Informació

Introducció:

Aquest procés bàsicament el que fa es rebre la consulta d'una base de dades en un document XML, l'analitza i la executa a SQL-Server. Genera els resultats a un fitxer en format XML.

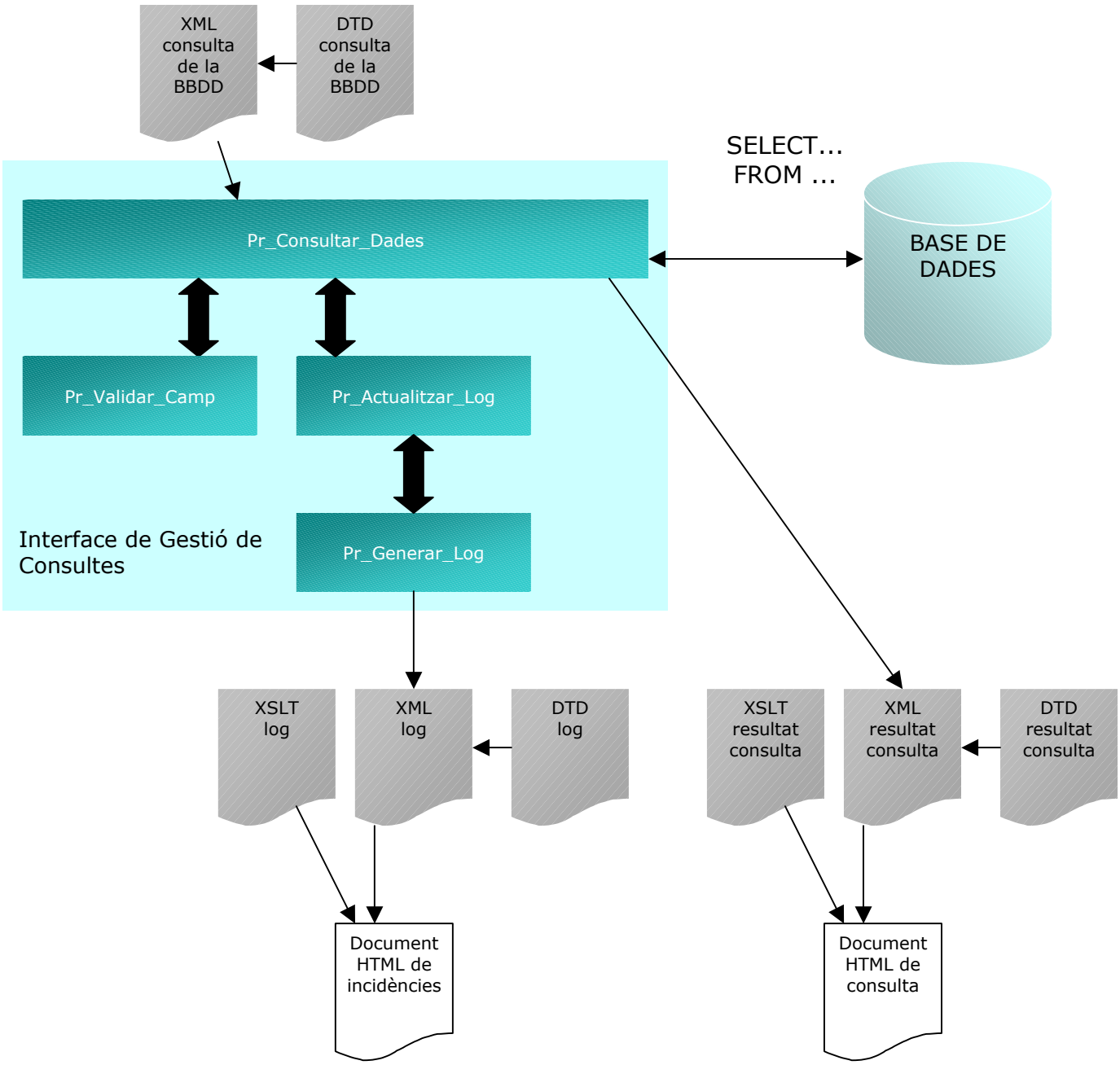


Figura 16 *Esquema Interface de Gestió de Consultes*

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Consultar_Dades:

Descripció: És l'encarregat de llançar les instruccions SQL necessàries per a extraure la informació sol·licitada. Obté la informació necessària d'un fitxer XML, utilitza el procediment Pr_Validar_Camp per verificar la condició de selecció, genera un fitxer XML amb els resultats de la consulta i crida al procediment Pr_Actualitzar_Log per generar un log d'incidències.

Paràmetres d'entrada: Document XML.

Paràmetres de sortida: Identificador consulta generada.

Crides a altres procediments:
sp_xml_preparedocument
sp_xml_removedocument
Pr_Validar_Camp
Pr_Actualitzar_Log

Pr_Validar_Camp:

Descripció: Procediment que s'utilitza per a verificar el contingut d'un camp d'una taula. Utilitza la consulta SYS_TAULES per obtenir les característiques d'una columna d'una taula. Si el resultat de la validació es erroni retorna un 0 i, en cas contrari, retorna 1, 2 o 3 en funció de si el camp es text, numèric o data.

Paràmetres d'entrada: Nom de la taula, nom del camp i valor.

Paràmetres de sortida: no n'hi han.

Crides a altres procediments:
Pr_Es_Numero
Pr_Es_Data

sp_xml_preparedocument:

Ja s'ha comentat a un apartat anterior.

sp_xml_removedocument:

Ja s'ha comentat a un apartat anterior.

Descripció dels documents XML utilitzats

XML Consulta Taula: S'utilitza per definir les dades que es volen obtenir de la Base de Dades relacional.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Estat SYSTEM "ConsultaDades.dtd">
<ConsultaDades>
        <Client>1</Client>
        <NomTaula>PERSONAL</NomTaula>
        <CampCondicio>
                <Nom>Id</Nom>
                <Valor>20</Valor>
                <Operador>'>'</Operador>
                <OperadorLogic>AND</OperadorLogic>
        </CampCondicio>
        <CampCondicio>
                <Nom>Nom</Nom>
                <Valor>Alejandro</Valor>
                <Operador>=</Operador>
                <OperadorLogic></OperadorLogic>
        </CampCondicio>
</ConsultaDades>
```

A l'arxiu XML s'especifiquen, en primer lloc, el client que vol efectuar la consulta i la taula sobre la que ho vol fer. Posteriorment hi han els tags <CampCondicio> on s'especifiquen els criteris de selecció dels registres. Dintre del tag <CampCondicio> s'especifica per una banda l'operador de comparació, tag <Operador>, i per un altra l'operador lògic per enllaçar-lo amb la següent condició, tag <OperadorLogic>.

La sentència SQL equivalent seria la següent:

*SELECT * FROM PERSONAL WHERE Id>20 AND Nom='Alejandro'*

XML resultat consulta: S'utilitza per mostrar quins han set els resultats de la consulta a la Base de Dades.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BDPUBLICA.dbo.PERSONAL>
<Id>30</Id>
<Nom>Alejandro</Nom>
<NIF>15856327M</NIF>
<Adreça>San Joan</Adreça>
<Data_Naixement>1992-11--1T00:00:00</Data_Naixement>
</BDPUBLICA.dbo.PERSONAL>
<Id>50</Id>
<Nom>Alejandro</Nom>
<NIF>85554327M</NIF>
<Adreça>San Pere</Adreça>
<Data_Naixement>1962-11--1T00:00:00</Data_Naixement>
</BDPUBLICA.dbo.PERSONAL>
```

A l'arxiu XML es poden veure els resultats de la consulta anterior. Es generen tants tags diferents com a camps hi té definits la taula seleccionada. A l'exemple anterior s'han obtingut dos registres amb nom igual a Alejandro i amb Id superior a 20.

1.3.9 Gestió d'errors

Introducció:

Sempre, després de qualsevol operació feta sobre la base de dades (alta, baixa, modificació, consulta...) es genera un registre a la taula de logs (TINLOG) i, a més a més, es genera un fitxer XML on el nom està composat pel usuari i pel nombre d'incidència. Aquest nombre d'incidència és el que retornen els diferents processos després de la seva execució. Per tant, quant l'usuari realitza, per exemple, una actualització de la base de dades, pot conèixer, després de la execució del procés, quin ha set el resultat recuperant el fitxer XML. D'aquesta forma l'usuari pot decidir si cal reenviar la informació un cop corregit l'error, la deixa com està ja que la incidència és mínima o bé no fa res per que les dades s'han actualitzat correctament.

Quant es detecta un error es comprova a la taula de definició d'incidències (TMADIN) si es tracta d'un error greu, d'un error lleu o d'informació. Només en el cas de que es tracti d'un error greu s'aturarà el procés. Si l'error es lleu el procés continua.

Descripció dels Procediments Emmagatzemats utilitzats:

Pr_Actualitzar_Log:

Descripció: Procediment que s'encarrega de obtenir la informació associada a la incidència passada de la taula TMADIN (definició d'incidències) tenint en compte l'idioma definit pel client. Actualitza la taula TINLOG (incidències) amb la informació

obtinguda i crida al procediment Pr_Generar_Log_XML per a que generi el fitxer XML corresponent.

Paràmetres d'entrada: Codi de Client, Número d'incidència i Nom del Procediment Emmagatzemat que el crida.

Paràmetres de sortida: Identificador de la incidència generada.

Crides a altres procediments:
Pr_Generar_Log_XML

Pr_Generar_Log_XML:

Descripció: Procediment que genera el fitxer XML relatiu a la incidència i el client passats. Aquest fitxer XML es genera mitjançant el codi de client i el identificador de la incidència.

Paràmetres d'entrada: Identificador d'incidència i Codi de Client.

Paràmetres de sortida: no n'hi ha.

Crides a altres procediments: no en fa.

Descripció dels documents XML utilitzats

XML log Incidències: S'utilitza per mostrar quins han set els resultats del procés de generació de taules. El diferents tags del fitxer coincideixen amb els camps de la taula relacional TINLOG.

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BDPUBLICA.dbo.TINLOG>
        <IDE_INC>9</IDE_INC>
        <COD_CLI>1</COD_CLI>
        <NOM_SPR>Pr_Crear_Taula</NOM_SPR>
        <DES_INC>Taula Generada
Correctament</DES_INC>
        <TIP_INC>Informacio</TIP_INC>
        <CTL_HOR>2004-04-12T15:47:51.170</CTL_HOR>
</BDPUBLICA.dbo.TINLOG>
```

Tots els arxius XML d'incidències presenten aquesta estructura ja que han set generats a partir de la taula interna de control d'incidències TINLOG. Al tag <IDE_INC> s'especifica un identificador única de la incidència, al tag <COD_CLI> el codi de client, al tag <NOM_SPR> el nom del Procediment Emmagatzemat que s'ha executat, al tag <DES_INC> la descripció de la incidència, al tag <TIP_INC> el tipus d'incidència i, finalment, al tag <CTL_HOR> l'hora i la data en que s'ha produït la incidència. És important comentar que els possibles tipus d'incidències son *Error Lleu*, *Error Greu* i *Informació*. Informació significa que el procediment emmagatzemat s'ha executat sense cap problema. Error Lleu indica que s'ha produït una petita incidència però que el procés ha continuat. En canvi, Error Greu confirma que s'ha produït un error que no ha permès la finalització del procediment.

DTD log Incidències: s'utilitza per validar el format del XML de Incidències.

XSLT log Incidències: aquest arxiu s'utilitza per donar-li una visualització HTML al fitxer XML d'Incidències. D'aquesta forma es poden oferir els resultats del procés d'una forma mes comprensible pel usuari.

1.4 Comentari de la gestió de XML que ofereix SQL-SERVER 2000

La gran acceptació del llenguatge XML per l'intercanvi d'informació ha fet que la gran majoria de Sistemes Gestors de Bases de Dades hagin anat incorporant eines que facilitin la gestió i generació d'aquestos documents. Microsoft no va ser la excepció i va incorporar diferents eines de gestió de documents XML a la versió 2000 de SQL-Server. Segurament la darrera versió d'aquest producte (v. 2005) haurà millorat i ampliat aquestes eines ja que la integració del XML, com a llenguatge de intercanvi de informació, no fa més que créixer.

A continuació es fa un breu comentari del funcionament d'aquestes eines que han set les utilitzades pels interfaces per gestionar i generar els arxius XML.

FOR XML

La instrucció FOR XML és pròpia d'SQL-Server i s'utilitza dintre del llenguatge SQL per especificar que els resultats obtinguts després d'una SELECT (instrucció estàndard de SQL per extraure informació d'una o varies taules de la base de dades) es mostraran com a un document vàlid XML. Aquesta instrucció té diferents paràmetres que permeten generar el arxiu XML de diferents formes. En concret els paràmetres utilitzats a aquest projecte han set AUTO i ELEMENTS. AUTO permet que cada taula de la clàusula FROM (part de la instrucció SELECT), amb al menys una columna a la clàusula SELECT, es representi com un element XML. ELEMENTS indica al generador XML que cada camp de la taula de la base de dades generi un tag diferent al arxiu XML.

OPENXML

La instrucció OPENXML també és pròpia d'SQL-Server i s'utilitza dintre del llenguatge SQL per interpretar un arxiu XML com si es tractes d'una taula o vista relacional més. Diferencia els diferents tags del arxiu i els interpreta com si es tractes de camps i registres de taules o vistes d'una base de dades relacional. Aquesta abstracció facilita molt la tasca del programador ja que no cal que es preocupi per gestionar les peculiaritats del arxiu XML.

1.5 Conclusions

Entenc que els principals objectius que es van plantejar a l'inici d'aquest projecte s'han assolit. S'han estudiat en profunditat les possibilitats que ofereix el llenguatge XML, i s'han utilitzat com a eina de intercanvi d'informació estàndard entre qualsevol aplicació externa i un Sistema de Gestió de Bases de Dades relacional. S'han generat diferents tipus d'arxiu XML que permeten comunicar a la aplicació quins tipus d'operacions es volen realitzar sobre la base de dades (creació de taules, modificació de dades, consultes...).

XML es una eina molt potent que facilita enormement la tasca d'intercanvi d'informació. És fàcil de processar i d'analitzar sintàcticament i inclou eines de validació de la informació que permeten simplificar aquesta tasca al programador. A més, inclou un mecanisme de representació visual que fa que un mateix arxiu XML es pugui visualitzar de diferents formes.

S'han desenvolupat diferents interfaces que són capaços d'interpretar els diferents arxius XML i de realitzar les operacions sol·licitades sobre la base de dades relacional. S'ha aconseguit que aquestos processos puguin generar taules senzilles a la base de dades. També poden fer insercions, modificacions o eliminacions de dades a les taules creades fen una sèrie de validacions pre-definides (comprovar que un camp existeix, verificar que la lletra d'un NIF és correcta, comprovar que una data és correcta...) Finalment, s'ha generat un interface que processa arxius XML de consulta de dades i retorna en format XML les dades obtingudes. Tots aquestos interfaces generen arxius XML de control (logs) que permeten verificar si la operació realitzada per l'interface ha tingut un resultat correcte o no.

Cal comentar que tots aquestos interfaces tenen bastants restriccions. Per exemple, a l'hora de crear una taula, no es permet camps de qualsevol tipus si no que s'accepten camps de tipus text, numèric o data. Això es així per dos motius, el primer per facilitar el desenvolupament de l'interface, i el segon per que la utilització de tipus de dades més específics pot dificultar la traducció als tipus de

dades de la base de dades relacional. Un altra restricció és, per exemple, que les consultes que es poden fer són sobre una única taula i no sobre conjunts de taules (no es permeten els JOINs).

En resum, els interfaces i els arxius XML és poden millorar i aquesta tasca podria ser l'objectiu de properes línies de treball. L'objectiu d'aquest projecte no era fer els millors interfaces possibles que fossin capaços d'interpretar qualsevol cas o arxius XML amb un gran nombre de possibilitats, si no demostrar que aquesta metodologia de treball era viable i podia donar uns resultats satisfactoris. Crec que això s'ha aconseguit i per tant com a conclusió final podem afirmar que, dintre del disseny d'aplicacions, és pot aconseguir un alt nivell d'abstracció de la base de dades relacional utilitzant mecanismes d'intercanvi d'informació com XML. Això és molt important, sobretot dintre del mon d'Internet, ja que permet que aplicacions totalment diferents utilitzin les dades d'un mateix nucli únicament generant i interpretant arxius XML. A més, aquesta independència permet que qui gestioni la base de dades pugui arribar a canviar el SGBD sense que la resta d'aplicacions se'n assabentin.

1.6 Futures línies de treball

A continuació es comenten algunes de les que podrien ser línies de treball que ampliessin aquest projecte.

- Traduir els interfaces a un altre llenguatge de programació i utilitzar un altre SGBD.

  Per desenvolupar aquest projecte s'ha utilitzat el gestor de bases de dades Microsfot SQL – SERVER. Dintre d'aquest i com a eina de programació, s'ha treballat amb procediments emmagatzemats. És podria utilitzar un altre SGBD com per exemple My-SQL (SGBD de codi obert) i desenvolupar els interfaces mitjançant, per exemple, servlets de Java que s'executessin sota un servidor de aplicacions com Apache Tomcat. Els servlets podrien processar el arxius XML enviats per altres aplicacions i generar les respostes també en arxius XML. El arxius XML serien els mateixos que els utilitzats en aquest projecte.

  Aquesta opció permetria mostrar com es pot canviar el gestor de bases de dades sense que les aplicacions que l'utilitzen se'n assabentin. És deslligaria el llenguatge de programació utilitzat (java) del SGBD i utilitzaria totalment eines lliures com són el propi java, My-SQL i Apache Tomcat.

- Dissenya diferents aplicacions amb tecnologies diferents (asp, jsp, wml ...) que utilitzin el mateix Servei d'Emmagatzemament de Dades.

  L'objectiu principal d'aquest projecte ha set implementar aquest servei, però únicament s'ha fet una petita prova amb aplicacions externes que l'utilitzin. Podria ser interessant, per mostrar la independència del servei, implementar diferents aplicacions amb tecnologies diferents que l'utilitzessin. També és podria aprofundir en el llenguatge XSLT que permet definit com s'ha de visualitzar un arxiu XML.

- Millorar les funcionalitats dels diferents interfaces.

  Tal i com s'ha comentat al apartat Conclusions l'objectiu d'aquest projecte no era fer uns interfaces que fossin capaços d'interpretar qualsevol cas o arxius XML amb un gran nombre de possibilitats, si no demostrar que aquesta

metodologia de treball era viable i podia donar uns resultats satisfactoris. Una possible línia d'ampliació d'aquest projecte seria ampliar les casuístiques a les que són capaços de donar resposta els interfaces. Per exemple, es podria dissenyar un estructura XML i un interface que fossin capaços de traduir consultes de dades complexes, entenent com a complexes consultes de varies taules amb JOIN, OTHER JOIN, agrupacions ...

Glossari

ADO: ActiveX® Data Objects és un conjunt d'objectes d'Automatització que utilitzen la API OLE DB y permeten que les aplicacions utilitzin dades d'orígens de dades OLE DB.

DTD (Document Type Definitions): Arxiu de definició que s'utilitza per validar arxius XML.

FOR XML: La instrucció FOR XML és pròpia d'SQL-Server i s'utilitza dintre del llenguatge SQL per especificar que els resultats obtinguts després d'una SELECT (instrucció estàndard de SQL per extraure informació d'una o varies taules de la base de dades) es mostraran com a un document vàlid XML

JDBC (Java Data Base Connectivity): Pont a diferents SGBD per a aplicacions desenvolupades en Java que permet l'accés a aquestos obviant les seves peculiaritats internes.

My-SQL: SGBD de codi obert. És un dels millors gestors de bases de dades lliures.

ODBC (Open Data Base Connectivity): Pont a diferents SGBD desenvolupat per Microsoft que permet l'accés a aquestos obviant les seves peculiaritats internes.

Oracle: és el SGBD més popular a tot el mon. En principi, sembla ser que és el millor gestor de bases de dades però, actualment, actualment hi ha d'altres que se li estan apropant.

PFC: Projecte Final de Carrera

Servei d'Emmagatzemament de Dades: servei que ofereix, dintre d'Internet, la possibilitat d'emmagatzemar dades de forma desatesa mitjançant l'intercanvi d'arxius XML.

Schema: Arxiu de definició que s'utilitza per validar arxius XML. És diferencia dels arxius DTD en que utilitza com a llenguatge el propi XML i permet especificar tipus de dades.

SGBD (Sistema de Gestió de Bases de Dades): Conjunt d'eines que faciliten la gestió de bases de dades relacionals.

SQL (Structured Query Language): Llenguatge estàndard de consulta a bases de dades relacionals.

SQL-SERVER: SGBD de Microsoft. Les seves darreres versions han aconseguit una important quota de mercat que fan que sigui un dels SGBD més utilitzats a l'actualitat.

XML (Extensible Markup Language): És un metallenguatge que ens permet definir llenguatges de marcat adequats per usos determinats. És un estàndard internacionalment reconegut que no pertany a cap empresa i la seva utilització és totalment lliure

XSL (Extended Stylesheet Language): És un llenguatge que permet definir una presentació o format per un document XML. Un mateix document XML pot tenir diferents fulles d'estil XSL que el mostren en diferents formats (HTML, WML, PDF, RTF...)

Bibliografia

- http://www.w3.org/TR/. Articles relacionats amb XML.
- Libros en Pantalla de Microsoft SQL-SERVER 2000
- Creación de Sitios WEB con SQL Server 2000. Autor: John Griffin. Editorial: Prentice Hall. ISBN:84-205-3462-5
- Introducción a XML. Autor: Doub Tidwell. http://www.ibm.com/developerWorks
- Introducción a XML en castellano. Autor: Alfredo Reino Romero. http://www.ibium.com/alf/xml/index.asp
- http://www.microsoft.com/sql/
- http://www.w3schools.com/xml/default.asp
- http://www.w3schools.com/xsl/default.asp
- http://www.w3schools.com/dtd/default.asp
- Apunts de l'assignatura Bases de Dades II de la Enginyeria en Informàtica de la UOC.
- Apunts de l'assignatura Comerç Electrònic de la Enginyeria en Informàtica de la UOC.
- Apunts de Metodologia i Gestió de Projectes Informàtica de la Enginyeria en Informàtica de la UOC.

Annexos

Codis fon de les eines de traspàs

Pr_Actualitzar_Log

```
CREATE PROCEDURE Pr_Actualitzar_Log
-- Procediment Emmagatzemat que genera una incidència al registre de incidències
-- i que crea un fitxer xml amb les principals dades de la incidència

-- Paràmetres
@CodCli int,
@NroInc int,
@NomSpr varchar(50)

AS

-- Variables locals
DECLARE @DesInc varchar(100)
DECLARE @TipInc varchar(20)
DECLARE @SelIdi int
DECLARE @IdeInc int

-- Obtenció del idioma
SELECT @SelIdi = (SELECT SEL_IDI FROM TMACLI WHERE COD_CLI=@CodCli)
-- Si no es Troba idioma per defecte és 1
SELECT @SelIdi = ISNULL(@SelIdi,1)

-- Obtenció de les característiques de la incidència
SELECT @DesInc = (SELECT DES_INC FROM TMADIN WHERE NRO_INC=@NroInc AND
SEL_IDI=@SelIdi)
SELECT @TipInc = (SELECT TIP_INC FROM TMADIN WHERE NRO_INC=@NroInc AND
SEL_IDI=@SelIdi)
SELECT @IdeInc = (SELECT COUNT(*) FROM TINLOG)

IF @IdeInc > 0
        BEGIN
                SELECT @IdeInc = (SELECT MAX(IDE_INC)+1 FROM TINLOG)
        END
ELSE
        BEGIN
                SELECT @IdeInc = 1
        END

-- Es crea el nou registre d'incidència
INSERT INTO TINLOG (IDE_INC,COD_CLI,NOM_SPR,DES_INC,TIP_INC) VALUES
(@IdeInc,@CodCli,@NomSpr,@DesInc,@TipInc)

-- Es genera el fitxer XML a la carpeta de logs
EXEC Pr_Generar_Log_XML @IdeInc,@CodCli

-- Retorna el identificador de la incidència
RETURN @IdeInc
```

Pr_Consultar_Dades

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE PROCEDURE Pr_Consultar_Dades
-- Procediment Emmagatzemat que gestiona les consultes a la base de dades.
-- Interpreta el fitxer xml passat i genera un fitxer xml amb les dades
-- sol.licitades

-- Paràmetres
@docXML varchar(1000),
@consultaXML varchar(1000) OUTPUT,
@IdeInc int OUTPUT

AS

-- Variables Locals
DECLARE @CodCli int, @idoc int, @Cliente int, @Longitud int, @Requerit int
DECLARE @SelIdi int
DECLARE @bRetorn int, @nError int, @nSysError int
DECLARE @DesInc varchar(100), @TipInc varchar(20)
DECLARE @docXML varchar(1000), @sAccio varchar(15)
DECLARE @NomTaula varchar(40), @Nom varchar(40), @Valor varchar (40)
DECLARE @Funcio varchar(100)
DECLARE @sqlString varchar(200), @sqlWhere varchar(1000), @sqlValors varchar(1000)
DECLARE @Columna varchar(50),@Tipus varchar(50),@Operador varchar(5)
DECLARE @OperadorLogic varchar(5), @sTmpOperador varchar(5)
DECLARE @sCmd varchar(1000), @sNomFic varchar(100), @sFecha varchar(20)
DECLARE @NomFic varchar(100)

SET @nError = 0

-- Representació Interna del Document XML.
EXEC  sp_xml_preparedocument @idoc OUTPUT, @docXML

-- Obtenció del client
SELECT @CodCli = (SELECT * FROM OPENXML(@idoc,'/ConsultaDades',2)
                  WITH (Client int))

-- Comprovació de que el client existeix
SELECT @Cliente = (SELECT COD_CLI FROM TMACLI WHERE COD_CLI=@CodCli)
IF ISNULL(@Cliente,0) = 0
BEGIN
        -- Eliminació del document de memòria
        EXEC sp_xml_removedocument @idoc
        -- Generació del log de incidències
        EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,4,'Pr_Consultar_Dades'
        RETURN
END

-- Obtenció del nom de la taula
SELECT @NomTaula = (SELECT * FROM OPENXML(@idoc,'/ConsultaDades',2)
                  WITH (NomTaula varchar(50)))

SET @sqlWhere = ''
SET @sqlValors = ''

-- OBTENCIÓ DE LA CLAUSULA WHERE PER A LA CONSULTA --
```

```
-- OBTENCIÓ DE LA CLAUSULA WHERE PER A LA CONSULTA --

-- Apertura d'un cursor al document XML
DECLARE Taula_Cursor CURSOR FOR
SELECT *
FROM OPENXML (@idoc, '/ConsultaDades/CampCondicio', 2) WITH (Nom varchar(50),
        Valor varchar(100), Operador varchar(5), OperadorLogic varchar(5))

OPEN Taula_Cursor

FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Operador,@OperadorLogic

WHILE @@FETCH_STATUS = 0
BEGIN
        SET @bRetorn = 1

        -- Validació del camp
        EXEC @bRetorn = Pr_Validar_Camp @NomTaula,@Nom,@Valor
        IF @bRetorn = 0
        BEGIN
                SET @nError = 12
                GOTO GestError
        END

        SET @sqlWhere = @sqlWhere + quotename( @Nom , '[') + @Operador

        -- Si es tracta d'un camp de tipus text o data s'afegeixen les cometes
        IF @bRetorn = 1 OR @bRetorn = 3
        BEGIN
                SET QUOTED_IDENTIFIER OFF
                SET @sqlWhere = @sqlWhere + quotename( @Valor ,"'")
                SET QUOTED_IDENTIFIER ON
        END
        ELSE
        BEGIN
                SET @sqlWhere = @sqlWhere + @Valor
        END

        -- S'afegeix l'operador lògic si hi ha més d'una condició de selecció
        SET @sTmpOperador = @OperadorLogic
        FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Operador,@OperadorLogic
        IF @@FETCH_STATUS = 0
        BEGIN
                SET @sqlWhere = @sqlWhere + ' ' + @sTmpOperador + ' '
        END
END

CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

-- EXECUCIÓ DE LA CONSULTA

SELECT @SelIdi = (SELECT SEL_IDI FROM TMACLI WHERE COD_CLI=@CodCli)
-- Si no es Troba idioma per defecte és 1
SELECT @SelIdi = ISNULL(@SelIdi,1)

-- S'obtenen les dades necessàries per la generació del fitxer
SELECT @DesInc = (SELECT DES_INC FROM TMADIN WHERE NRO_INC=14 AND SEL_IDI=@SelIdi)
SELECT @TipInc = (SELECT TIP_INC FROM TMADIN WHERE NRO_INC=14 AND SEL_IDI=@SelIdi)
SELECT @IdeInc = (SELECT COUNT(*) FROM TINLOG)

IF @IdeInc > 0
        BEGIN
                SELECT @IdeInc = (SELECT MAX(IDE_INC)+1 FROM TINLOG)
        END
ELSE
        BEGIN
                SELECT @IdeInc = 1
        END
```

```
-- Es crea el nou registre d'incidència (com a consulta)
INSERT INTO TINLOG (IDE_INC,COD_CLI,NOM_SPR,DES_INC,TIP_INC) VALUES
(@IdeInc,@CodCli,'Pr_Consultar_Dades',@DesInc,@TipInc)

SELECT @NomFic = CAST(@CodCli as varchar) + '_' + CAST(@IdeInc as varchar)

-- S'exporten les dades al fitxer
SELECT @sqlString = 'master..xp_cmdshell' +  quotename('bcp "SELECT * FROM BDPUBLICA.dbo.'
+ @NomTaula + ' WHERE ' + @sqlWhere + ' FOR XML AUTO,ELEMENTS" queryout
c:\inetpub\pfc\logs\' + @NomFic + '.xml -c',"]")
SET QUOTED_IDENTIFIER OFF
SET @sCmd = @sqlString
SELECT @sCmd
EXEC (@sCmd)
SET QUOTED_IDENTIFIER ON

-- Control d'errors
SET @nSysError = @@ERROR
IF (@nSysError <> 0)
BEGIN
        SET @nError = 9
        GOTO GestError2
END

-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,13,'Pr_Consultar_Dades'
RETURN

-- Gestió d'errors greus
GestError:

-- Es tanquen els cursors
CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

GestError2:
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,@nError,'Pr_Consultar_Dades'

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Crear_Taula

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE    PROCEDURE Pr_Crear_Taula
-- Procediment Emmagatzemat que gestiona la creació d'una taula a la base de
-- dades a partir d'un document XML passat
```

```
-- Paràmetres
@docXML varchar(1000),
@IdeInc int OUTPUT
AS

-- Variables locals
DECLARE @idoc int
DECLARE @Clau int
DECLARE @sqlString varchar(1000)
DECLARE @sqlCamps varchar(1000)
DECLARE @sqlCampsClau varchar(1000)
DECLARE @Nom varchar(50)
DECLARE @NomTaula varchar(100)
DECLARE @Tipus varchar(20)
DECLARE @CodCli int
DECLARE @Cliente int
DECLARE @Retorn int
DECLARE @docXML varchar(1000)

-- Representació interna del document XML
EXEC  sp_xml_preparedocument @idoc OUTPUT, @docXML

-- Obtenció del client
SELECT @CodCli = (SELECT * FROM OPENXML(@idoc,'/DefTaula',2)
WITH (Client int))

-- Comprovació de que el client existeix
SELECT @Cliente = (SELECT COD_CLI FROM TMACLI WHERE COD_CLI=@CodCli)
IF ISNULL(@Cliente,0) = 0
BEGIN
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc
-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,4,'Pr_Crear_Taula'
RETURN
END

-- Obtenció del nom de la taula
SELECT @NomTaula = (SELECT * FROM OPENXML(@idoc,'/DefTaula',2)
WITH (NomTaula varchar(50)))

-- Obtenció dels camps a generar a la taula
DECLARE Taula_Cursor CURSOR FOR
SELECT *
FROM OPENXML (@idoc, '/DefTaula/Camp', 2)
WITH (Nom varchar(50),
Tipus varchar(20),
Clau integer)

OPEN Taula_Cursor

SELECT @sqlString = ''
SELECT @sqlCamps = ''
SELECT @sqlCampsClau = ''

FETCH NEXT FROM Taula_Cursor INTO @Nom,@Tipus,@Clau
WHILE @@FETCH_STATUS = 0
BEGIN
-- Assignació del tipus de camp
SELECT @sqlCamps = @sqlCamps + quotename( @Nom , '[')
IF @Tipus = 'Numèric'
BEGIN
SELECT @sqlCamps = @sqlCamps + ' [numeric](18,0)'
END
IF @Tipus = 'Text'
BEGIN
SELECT @sqlCamps = @sqlCamps + ' [nvarchar](50)'
END
```

```
        IF @Tipus = 'Data'
        BEGIN
                SELECT @sqlCamps = @sqlCamps + ' [datetime]'
        END

        -- Contro de camps clau
        IF @Clau = 1
        BEGIN
                SELECT @sqlCamps = @sqlCamps + ' NOT NULL'
                IF @sqlCampsClau <> ''
                BEGIN
                        SELECT @sqlCampsClau = @sqlCampsClau + ','
                END
                SELECT @sqlCampsClau = @sqlCampsClau + @Nom
        END

        FETCH NEXT FROM Taula_Cursor INTO @Nom,@Tipus,@Clau

        IF @@FETCH_STATUS = 0
        BEGIN
                SELECT @sqlCamps = @sqlCamps + ','
        END
END

CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació de la instrucció sql per crear la taula
SELECT @sqlString='CREATE TABLE ' + quotename( @NomTaula , '[') + '(' + @sqlCamps

IF @sqlCampsClau <> ''
BEGIN
        SELECT @sqlString = @sqlString + ', PRIMARY KEY(' + @sqlCampsClau + ')'
END
SELECT @sqlString = @sqlString + ')'

EXEC (@sqlString)

-- Control d'errors
IF (@@ERROR <> 0)
BEGIN
        EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,5,'Pr_Crear_Taula'
        RETURN
END

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,3,'Pr_Crear_Taula'


GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Es_Data

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO


CREATE  PROCEDURE Pr_Es_Data
-- Procediment emmagatzemat que retorna 0 o 1 en funció de si el valor passat
-- correspon a una data ben construïda o no
@valor varchar(1000)
AS
DECLARE @bReturn as int
DECLARE @nNum as FLOAT

SET @bReturn = ISDATE(@valor)

RETURN @bReturn

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Es_Numero

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO


CREATE  PROCEDURE Pr_Es_Numero
-- Procediment emmagatzemat que retorna 0 o 1 en funció de si el valor passat
-- correspon a un valor numèric

-- Paràmetres
@valor varchar(1000)

AS

-- Variables locals
DECLARE @bReturn as int
DECLARE @nNum as FLOAT

SET @bReturn = ISNUMERIC(@valor)

RETURN @bReturn

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Generar_Log_XML

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO


CREATE   PROCEDURE Pr_Generar_Log_XML
-- Procediment Emmagatzemat que genera un fitxer XML d'incidències a partir
-- d'una incidència que figura al registre d'incidències (taula TINLOG)

-- Paràmetres
@IdeInc as int,
@CodCli as int

AS

-- Variables Locals
DECLARE @NomFic varchar(100)
DECLARE @sSentencia varchar(1000)

SELECT @NomFic = CAST(@CodCli as varchar) + '_' + CAST(@IdeInc as varchar)

SELECT @sSentencia = 'master..xp_cmdshell' + quotename('bcp "SELECT * FROM
BDPUBLICA.dbo.TINLOG WHERE IDE_INC=' + CAST(@IdeInc as varchar) + ' FOR XML
AUTO,ELEMENTS" queryout c:\inetpub\pfc\logs\' + @NomFic + '.xml -c')

EXEC (@sSentencia)


GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Gestio_Altes

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE       PROCEDURE Pr_Gestio_Altes
-- Procediment Emmagatzemat que inserta noves files a les taules de la base
-- de dades en funció del fitxer XML passat. Verifica que les dades siguin
-- correctes, que els valors obligatoris hi siguin i, si s'han especificat,
-- efectua les funcions de validació.

-- Paràmetres
@docXML varchar(1000),
@IdeInc int OUTPUT

AS

-- Variables locals
DECLARE @CodCli int, @idoc int, @Cliente int, @Longitud int, @Requerit int
DECLARE @bRetorn int, @nError int, @nSysError int
DECLARE @docXML varchar(1000), @sAccio varchar(15), @sProc varchar(50)
DECLARE @NomTaula varchar(40), @Nom varchar(40), @Valor varchar (40)
DECLARE @sqlString varchar(200), @sqlCamps varchar(1000), @sqlValors varchar(1000)
```

```
DECLARE @Funcio varchar(40),@TipInc varchar(20)
DECLARE @Columna varchar(50),@Tipus varchar(50)

SET @nError = 0

-- Representació interna del document XML
EXEC  sp_xml_preparedocument @idoc OUTPUT, @docXML

-- Obtenció del client
SELECT @CodCli = (SELECT * FROM OPENXML(@idoc,'/AltaDades',2)
                  WITH (Client int))

-- Comprovació de que el client existeix
SELECT @Cliente = (SELECT COD_CLI FROM TMACLI WHERE COD_CLI=@CodCli)
IF ISNULL(@Cliente,0) = 0
BEGIN
        -- Eliminació del document de memòria
        EXEC sp_xml_removedocument @idoc
        -- Generació del log de incidències
        EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,4,'Pr_Gestio_Altes'
        RETURN
END

-- Obtenció del nom de la taula
SELECT @NomTaula = (SELECT * FROM OPENXML(@idoc,'/AltaDades',2)
                  WITH (NomTaula varchar(50)))

SET @sqlCamps = ''
SET @sqlValors = ''

-- S'obtenen els valors de les columnes corresponents a la taula
DECLARE Taula_Mod CURSOR FOR
SELECT COLUMNA,TIPUS,LONGITUD,REQUERIT
FROM SYS_TAULES WHERE NOM=@NomTaula
ORDER BY POSICIO ASC

OPEN Taula_Mod

FETCH NEXT FROM Taula_Mod INTO @Columna,@Tipus,@Longitud,@Requerit
WHILE @@FETCH_STATUS = 0
BEGIN
        -- Apertura d'un cursor al document XML
        DECLARE Taula_Cursor CURSOR FOR
        SELECT *
        FROM OPENXML (@idoc, '/AltaDades/Camp', 2) WITH (Nom varchar(50),
                Valor varchar(100),
                Funcio varchar(100))
                WHERE Nom=@Columna

        OPEN Taula_Cursor

        FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Funcio

        -- SI EL CAMP S'HA ESPECIFICAT
        IF @@FETCH_STATUS = 0
        BEGIN
                SET @bRetorn = 1

                -- Validació del tipus
                IF @Tipus = 'numeric'
                BEGIN
                        EXEC @bRetorn = Pr_Es_Numero @Valor
                        IF @bRetorn = 0
                        BEGIN
                                SET @nError = 7
                                GOTO GestError
                        END
```

```
                END
                IF @Tipus = 'datetime'
                BEGIN
                        EXEC @bRetorn = Pr_Es_Data @Valor
                        IF @bRetorn = 0
                        BEGIN
                                SET @nError = 8
                                GOTO GestError
                        END
                END

                SET @sqlCamps = @sqlCamps + quotename( @Nom , '[')

                IF @Tipus <> 'numeric'
                BEGIN
                        SET @sqlValors = @sqlValors + quotename( @Valor , '"')
                END
                ELSE
                BEGIN
                        SET @sqlValors = @sqlValors + @Valor
                END

                FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Funcio
        END

        -- SI EL CAMP NO S'HA ESPECIFICAT
        ELSE
        BEGIN
                -- Si el camp no s'ha definit i es requereix es mostra error
                IF @Requerit = 0
                BEGIN
                        SET @nError = 6
                        GOTO GestError
                END
                ELSE
                BEGIN
                        -- Si no s'ha especificat valor però no es requerit s'actualitzarà
                        -- com a NULL
                        SET @sqlCamps = @sqlCamps + quotename( @Columna , '[')
                        SET @sqlValors = @sqlValors + 'Null'
                END
        END

        -- APLICACIÓ DE LA FUNCIÓ DE VALIDACIÓ AL CAMP
        -- Si s'ha especificat una funció de validació aquesta s'aplica
        IF LEN(@Funcio) > 0
        BEGIN
                -- Es crida al procediment Emmagatzemat de validació de funcions
                EXEC @bRetorn = Pr_Validar_Funcio @Funcio,@Valor
                IF @bRetorn > 0
                BEGIN
                        SELECT @TipInc = (SELECT TIP_INC FROM TMADIN WHERE
                                NRO_INC=@bRetorn AND SEL_IDI=1)

                        -- Gestió del Tipus d'error.
                        IF @TipInc = 'Error Greu'
                        BEGIN
                                --  Si es greu finalitza el procés
                                SET @nError = @bRetorn
                                GOTO GestError
                        END
                        ELSE
                        BEGIN
                                -- Si es lleu es genera una incidència i continua el procés
                                EXEC @IdeInc = Pr_Actualitzar_Log
@CodCli,@bRetorn,'Pr_Gestio_Altes'
                        END
                END
        END
```

```
        CLOSE Taula_Cursor
        DEALLOCATE Taula_Cursor

        FETCH NEXT FROM Taula_Mod INTO @Columna,@Tipus,@Longitud,@Requerit

        IF @@FETCH_STATUS = 0
        BEGIN
                SET @sqlCamps = @sqlCamps + ','
                SET @sqlValors = @sqlValors + ','
        END
END

CLOSE Taula_Mod
DEALLOCATE Taula_Mod

SET QUOTED_IDENTIFIER OFF
SELECT @sqlString='INSERT INTO ' + @NomTaula + ' (' + @sqlCamps + ') VALUES (' + @sqlValors
+ ')'
EXEC (@sqlString)

-- Control d'errors
SET @nSysError = @@ERROR
IF (@nSysError <> 0)
BEGIN
        -- Error per clau duplicada
        IF @nSysError = 2627
        BEGIN
                SET @nError = 10
                GOTO GestError2
        END
        SET @nError = 9
        GOTO GestError2
END
SET QUOTED_IDENTIFIER ON

-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,11,'Pr_Gestio_Altes'
RETURN

-- Gestió d'errors greus
GestError:

-- Es tanquen els cursors
CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor
CLOSE Taula_Mod
DEALLOCATE Taula_Mod

GestError2:
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,@nError,'Pr_Gestio_Altes'


GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Gestio_Baixes

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO


CREATE  PROCEDURE Pr_Gestio_Baixes
@docXML varchar(1000),
@IdeInc int OUTPUT
AS
DECLARE @CodCli int, @idoc int, @Cliente int, @Longitud int, @Requerit int
DECLARE @bRetorn int, @nError int, @nSysError int
DECLARE @docXML varchar(1000), @sAccio varchar(15)
DECLARE @NomTaula varchar(40), @Nom varchar(40), @Valor varchar (40)
DECLARE @Funcio varchar(100)
DECLARE @sqlString varchar(200), @sqlWhere varchar(1000)
DECLARE @Columna varchar(50),@Tipus varchar(50)

SET @nError = 0

-- Representació interna del document XML
EXEC  sp_xml_preparedocument @idoc OUTPUT, @docXML

-- Obtenció del client
SELECT @CodCli = (SELECT * FROM OPENXML(@idoc,'/BaixaDades',2)
                  WITH (Client int))

-- Comprovació de que el client existeix
SELECT @Cliente = (SELECT COD_CLI FROM TMACLI WHERE COD_CLI=@CodCli)
IF ISNULL(@Cliente,0) = 0
        BEGIN
                -- Eliminació del document de memòria
                EXEC sp_xml_removedocument @idoc
                -- Generació del log de incidències
                EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,4,'Pr_Gestio_Baixes'
                RETURN
        END

-- Obtenció del nom de la taula
SELECT @NomTaula = (SELECT * FROM OPENXML(@idoc,'/BaixaDades',2)
                  WITH (NomTaula varchar(50)))

SET @sqlWhere = ''

-- OBTENCIÓ DE LA CLAUSULA WHERE PER A LA MODIFICACIÓ --

-- Apertura d'un cursor al document XML
DECLARE Taula_Cursor CURSOR FOR
SELECT *
FROM OPENXML (@idoc, '/BaixaDades/Camp', 2) WITH (Nom varchar(50),
        Valor varchar(100))

OPEN Taula_Cursor

FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor

WHILE @@FETCH_STATUS = 0
BEGIN
        SET @bRetorn = 1

        -- Validació del camp
        EXEC @bRetorn = Pr_Validar_Camp @NomTaula,@Nom,@Valor
        IF @bRetorn = 0
        BEGIN
                SET @nError = 12
                GOTO GestError
        END
```

```
		SET @sqlWhere = @sqlWhere + quotename( @Nom , '[') + '='
		-- Si es tracta d'un camp de tipus text o data s'afegeixen les cometes
		IF @bRetorn = 1 OR @bRetorn = 3
		BEGIN
			SET @sqlWhere = @sqlWhere + quotename( @Valor , '"')
		END
		ELSE
		BEGIN
			SET @sqlWhere = @sqlWhere + @Valor
		END

		FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor
		IF @@FETCH_STATUS = 0
		BEGIN
			SET @sqlWhere = @sqlWhere + ' AND '
		END
END

CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

SELECT @sqlWhere

-- EXECUCIÓ DE LA BAIXA

SET QUOTED_IDENTIFIER OFF
SELECT @sqlString='DELETE ' + @NomTaula + ' WHERE ' + @sqlWhere
EXEC (@sqlString)
-- Control d'errors
SET @nSysError = @@ERROR
IF (@nSysError <> 0)
BEGIN
		-- Error per clau duplicada
		IF @nSysError = 2627
		BEGIN
			SET @nError = 10
			GOTO GestError2
		END
			SET @nError = 9
		GOTO GestError2
END
SET QUOTED_IDENTIFIER ON

-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,11,'Pr_Gestio_Baixes'
RETURN

-- Gestió d'errors greus
GestError:

-- Es tanquen els cursors
CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

GestError2:
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,@nError,'Pr_Gestio_Baixes'

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Gestio_Modificacions

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO



CREATE      PROCEDURE Pr_Gestio_Modificacions
-- Procediment Emmagatzemat que modifica les files a les taules de la base
-- de dades en funció del fitxer XML passat. Verifica que les dades siguin
-- correctes, que els valors obligatoris hi siguin i, si s'han especificat,
-- efectua les funcions de validació.

-- Paràmetres
@docXML varchar(1000),
@IdeInc int OUTPUT

AS

-- Variables locals
DECLARE @CodCli int, @idoc int, @Cliente int, @Longitud int, @Requerit int
DECLARE @bRetorn int, @nError int, @nSysError int
DECLARE @docXML varchar(1000), @sAccio varchar(15)
DECLARE @NomTaula varchar(40), @Nom varchar(40), @Valor varchar (40)
DECLARE @Funcio varchar(100)
DECLARE @sqlString varchar(200), @sqlWhere varchar(1000), @sqlValors varchar(1000)
DECLARE @Columna varchar(50),@Tipus varchar(50),@Operador varchar(5)
DECLARE @OperadorLogic varchar(5), @sTmpOperador varchar(5),@TipInc varchar(20)

SET @nError = 0

-- Representació interna del document XML
EXEC  sp_xml_preparedocument @idoc OUTPUT, @docXML

-- Obtenció del client
SELECT @CodCli = (SELECT * FROM OPENXML(@idoc,'/ModDades',2)
                  WITH (Client int))

-- Comprovació de que el client existeix
SELECT @Cliente = (SELECT COD_CLI FROM TMACLI WHERE COD_CLI=@CodCli)
IF ISNULL(@Cliente,0) = 0
BEGIN
          -- Eliminació del document de memòria
          EXEC sp_xml_removedocument @idoc
          -- Generació del log de incidències
          EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,4,'Pr_Gestio_Modificacions'
          RETURN
END

-- Obtenció del nom de la taula
SELECT @NomTaula = (SELECT * FROM OPENXML(@idoc,'/ModDades',2)
                  WITH (NomTaula varchar(50)))

SET @sqlWhere = ''
SET @sqlValors = ''

-- OBTENCIÓ DE LA CLAUSULA WHERE PER A LA MODIFICACIÓ --

-- Apertura d'un cursor al document XML
DECLARE Taula_Cursor CURSOR FOR
SELECT *
FROM OPENXML (@idoc, '/ModDades/CampCondicio', 2) WITH (Nom varchar(50),
          Valor varchar(100), Operador varchar(5), OperadorLogic varchar(5))
```

```
OPEN Taula_Cursor

FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Operador,@OperadorLogic

WHILE @@FETCH_STATUS = 0
BEGIN
	SET @bRetorn = 1

	-- Validació del camp
	EXEC @bRetorn = Pr_Validar_Camp @NomTaula,@Nom,@Valor
	IF @bRetorn = 0
	BEGIN
		SET @nError = 12
		GOTO GestError
	END

	SET @sqlWhere = @sqlWhere + quotename( @Nom , '[') + @Operador
	-- Si es tracta d'un camp de tipus text o data s'afegeixen les cometes
	IF @bRetorn = 1 OR @bRetorn = 3
	BEGIN
		SET @sqlWhere = @sqlWhere + quotename( @Valor , '''')
	END
	ELSE
	BEGIN
		SET @sqlWhere = @sqlWhere + @Valor
	END

	SET @sTmpOperador = @OperadorLogic
	FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Operador,@OperadorLogic
	IF @@FETCH_STATUS = 0
	BEGIN
		SET @sqlWhere = @sqlWhere + ' ' + @sTmpOperador + ' '
	END
END

CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

-- OBTENCIÓ DELS VALORS A MODIFICAR

-- Apertura d'un cursor al document XML
DECLARE Taula_Cursor CURSOR FOR
SELECT *
FROM OPENXML (@idoc, '/ModDades/CampMod', 2) WITH (Nom varchar(50),
	Valor varchar(100), Funcio varchar(100))

OPEN Taula_Cursor

FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Funcio

WHILE @@FETCH_STATUS = 0
BEGIN
	SET @bRetorn = 1

	-- Validació del camp
	EXEC @bRetorn = Pr_Validar_Camp @NomTaula,@Nom,@Valor
	IF @bRetorn = 0
	BEGIN
		SET @nError = 12
		GOTO GestError
	END

	SET @sqlValors = @sqlValors + quotename( @Nom , '[') + '='
	-- Si es tracta d'un camp de tipus text o data s'afegeixen les cometes
	IF @bRetorn = 1 OR @bRetorn = 3
	BEGIN
		SET @sqlValors = @sqlValors + quotename( @Valor , '''')
	END
```

```
        ELSE
        BEGIN
                SET @sqlValors = @sqlValors + @Valor
        END

        -- APLICACIÓ DE LA FUNCIÓ DE VALIDACIÓ AL CAMP

        IF LEN(@Funcio) > 0
        BEGIN
                -- Es crida al procediment Emmagatzemat de validació de funcions
                EXEC @bRetorn = Pr_Validar_Funcio @Funcio,@Valor
                IF @bRetorn > 0
                BEGIN
                        SELECT @TipInc = (SELECT TIP_INC FROM TMADIN WHERE
                                NRO_INC=@bRetorn AND SEL_IDI=1)

                        -- Gestió del Tipus d'error.
                        IF @TipInc = 'Error Greu'
                        BEGIN
                                --  Si es greu finalitza el procés
                                SET @nError = @bRetorn
                                GOTO GestError
                        END
                        ELSE
                        BEGIN
                                -- Si es lleu es genera una incidència i continua el procés
                                EXEC @IdeInc = Pr_Actualitzar_Log
@CodCli,@bRetorn,'Pr_Gestio_Modificacions'
                        END
                END
        END

        FETCH NEXT FROM Taula_Cursor INTO @Nom,@Valor,@Funcio
        IF @@FETCH_STATUS = 0
        BEGIN
                SET @sqlValors = @sqlValors + ','
        END
END

CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

SELECT @sqlWhere
SELECT @sqlValors

-- EXECUCIÓ DE LA MODIFICACIÓ

SET QUOTED_IDENTIFIER OFF
SELECT @sqlString='UPDATE ' + @NomTaula + ' SET ' + @sqlValors + ' WHERE ' + @sqlWhere
EXEC (@sqlString)
-- Control d'errors
SET @nSysError = @@ERROR
IF (@nSysError <> 0)

BEGIN
        -- Error per clau duplicada
        IF @nSysError = 2627
        BEGIN
                SET @nError = 10
                GOTO GestError2
        END
        SET @nError = 9
        GOTO GestError2
END
SET QUOTED_IDENTIFIER ON
```

```
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,11,'Pr_Gestio_Modificacions'
RETURN

-- Gestió d'errors greus
GestError:

-- Es tanquen els cursors
CLOSE Taula_Cursor
DEALLOCATE Taula_Cursor

GestError2:
-- Eliminació del document de memòria
EXEC sp_xml_removedocument @idoc

-- Generació del log de incidències
EXEC @IdeInc = Pr_Actualitzar_Log @CodCli,@nError,'Pr_Gestio_Modificacions'



GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Validar_Camp

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE   PROCEDURE Pr_Validar_Camp
-- Procediment Emmagatzemat que s'utilitza per a verificar el contigut d'un camp
-- d'una taula. Retorna 0 si el contingut és erroni i 1, 2 o 3 en el cas de que
-- sigui correcte

-- Paràmetres
@sNomTaula varchar(50),
@sNomCamp varchar(50),
@sValor varchar(50)

AS

-- Variables locals
DECLARE @Tipus varchar(20), @Longitud int, @Requerit int, @Posicio int
DECLARE @bRetorn int

-- Obtenció de les característiques del camp de la taula
DECLARE Cursor_Validar_Camp CURSOR FOR
SELECT TIPUS,LONGITUD,REQUERIT,POSICIO
FROM SYS_TAULES WHERE NOM=@sNomTaula AND COLUMNA=@sNomCamp

OPEN Cursor_Validar_Camp

FETCH NEXT FROM Cursor_Validar_Camp INTO @Tipus,@Longitud,@Requerit,@Posicio
WHILE @@FETCH_STATUS = 0
BEGIN
```

```
        -- Comprovació del tipus de camp
        IF @Tipus = 'numeric'
        BEGIN
                EXEC @bRetorn = Pr_Es_Numero @sValor
        END
        IF @Tipus = 'datetime'
        BEGIN
                EXEC @bRetorn = Pr_Es_Data @sValor
        END

        FETCH NEXT FROM Cursor_Validar_Camp INTO @Tipus,@Longitud,@Requerit,@Posicio
END

CLOSE Cursor_Validar_Camp
DEALLOCATE Cursor_Validar_Camp

IF @bRetorn = 0
        BEGIN
                -- CAMP INCORRECTE
                RETURN 0
        END
ELSE
        BEGIN
                -- CAMP CORRECTE
                IF @Tipus = 'nvarchar'
                BEGIN
                        RETURN 1
                END
                IF @Tipus = 'numeric'
                BEGIN
                        RETURN 2
                END
                IF @Tipus = 'datetime'
                BEGIN
                        RETURN 3
                END
        END


GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Pr_Validar_Funcio

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS OFF
GO

CREATE   PROCEDURE Pr_Validar_Funcio
-- Procediment Emmagatzemat que executa el procés de validació assignat per
-- un camp. Retorna 0 si el procés ha set correcte i el número d'incidència si
-- ha set incorrecte

-- Paràmetres
@sFuncio varchar(50),
@sVal varchar(50)

AS

-- Variables locals
DECLARE @bRetorn int, @NroInc int
```

```
-- S'obté el Procediment emmagatzemat de la taula de definició de
-- funcions de validació
SELECT @NroInc = (SELECT NRO_INC FROM TMADFU WHERE NOM_FUN=@sFuncio)

IF ISNULL(@NroInc,0) > 0
BEGIN

        -- Es crida al procediment emmagatzemat corresponent i en cas
        -- de que la validació no sigui correcta es retorna el número de
        -- incidència
        IF @sFuncio = 'Validar_NIF'
        BEGIN
                EXEC @bRetorn = Pr_Validar_NIF @sVal
        END

        IF @bRetorn = 1
        BEGIN
                RETURN 0
        END
        ELSE
        BEGIN
                RETURN @NroInc
        END
END
```

Pr_Validar_NIF

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE   PROCEDURE Pr_Validar_Nif
-- Procediment Emmagatzemat que valida un NIF. Retorna -1 si és incorrecte i
-- 1 si és correcte

-- Paràmetres
@strA varchar(15)

AS

-- Variables locals
DECLARE @cCADENA varchar(20), @sCaracter varchar(20)
DECLARE @cNUMEROS varchar(10)
DECLARE @strT varchar(15), @strB varchar(15)
DECLARE @a int, @b int, @i int, @c int
DECLARE @NIF numeric

SET @cCADENA =  'TRWAGMYFPDXBNJZSQVHLCKE'
SET @cNUMEROS = '0123456789'

SET @strT = RTRIM(LTRIM(@strA))

IF LEN(@strT) = 0
BEGIN
        RETURN -1
END

SET @strB = ''
SET @sCaracter = ''
SET @i = 1
```

```
-- S'eliminen els caràcters no numèrics
WHILE @i <= LEN (@strA)
BEGIN
	IF ISNUMERIC(SUBSTRING(@strA,@i,1)) = 1
	BEGIN
		SET @strB = @strB + SUBSTRING(@strA,@i,1)
	END
	ELSE
	BEGIN
		SET @sCaracter = @sCaracter + SUBSTRING(@strA,@i,1)
	END

	SET @i = @i + 1
END
-- Si s'ha indicat al NIF més de un caràcter no numèric es retorna error
IF LEN(@sCaracter) > 1
BEGIN
	RETURN -1
END

SET @strA = @strB
SET @a = 0
SET @NIF = CAST(@strA as numeric)

SET @b = CAST((@NIF/24) as int)
SET @c = @NIF - (24 * @b)
SET @a = @a + @c
SET @NIF = @b

WHILE @b <> 0
BEGIN
	SET @b = CAST((@NIF/24) as int)
	SET @c = @NIF - (24 * @b)
	SET @a = @a + @c
	SET @NIF = @b
END

SET @b = CAST((@a / 23) as int)
SET @c = @a - (23 * @b)

IF (SUBSTRING(@cCADENA,@c+1,1)) <> @sCaracter
BEGIN
	RETURN -1
END
ELSE
BEGIN
	RETURN 1
END

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```