UNIVERSITAT OBERTA DE CATALUNYA

Enginyeria Tècnica en Informàtica de Gestió

Treball de Final de Carrera (Àrea de Bases de dades):

Comparació entre els llenguatges
XQuery* del *W3C* i *X-Query* de *Tamino

Alumne: Carlos Manuel Martí Hernández
Direcció: Antoni Pérez Navarro

Curs: 2003-04 (segon semestre)

Comparació entre els llenguatges *XQuery* del *W3C* i *X-Query* de *Tamino*

Resum:

L'*XML* (*eXtensible Markup Language*) va ser definit el 1998 pel *W3C* (*World Wide Web Consortium*) per a representar informació de manera independent a qualsevol format. Per tal d'accedir a aquesta informació, d'extreure-la i de manipular-la, el *W3C* està treballant en la definició d'un llenguatge de consultes per a *XML* anomenat *XQuery*. L'emmagatzemament i el tractament d'aquesta informació pot gestionar-se des d'algunes bases de dades relacionals convencionals (com ara *Oracle*, des de la versió *8i*). Tanmateix, la solució òptima consisteix en gestionar aquestes dades en el seu format nadiu, sense haver de recórrer a cap conversió de formats, mantenint intacta l'estructura dels documents. L'empresa *Software AG* distribueix un sistema nadiu, *Tamino XML Server* (actualment en la versió 4.1.4.1), que permet, a més, treballar amb dades *XML* amb dos llenguatges de consultes alternatius sobre dades *XML*. D'una banda, Tamino proporciona una implementació de l'especificació *XQuery* del *W3C* (*Tamino XQuery 4*), i d'una altra, ofereix *X-Query*, un llenguatge propi de *Tamino*. Per mitjà de l'estudi previ d'aquestes tecnologies, i del plantejament i de la resolució ulteriors d'un cas pràctic, es fa una comparativa entre les possibilitats que ofereixen l'*XQuery* (sense guió) del *W3C* i l'*X-Query* (amb guió) de *Tamino*, dins del sistema de bases de dades nadiu de *Software AG*. La majoria de les funcionalitats que proporcionen *XQuery* i *X-Query* són comunes als dos llenguatges, i poden obtenir-se, o bé seguint el mateix procediment en ambdós casos, o bé fent ús de petites variacions sintàctiques. En *XQuery*, en general, és possible d'expressar consultes més complexes que no pas en *X-Query*, les quals poden abarcar gairebé qualsevol necessitat. L*'X-Query*, en canvi, comporta una simplificació en la sintaxi, en relació a la de l'*XQuery*, la qual cosa no sempre implica una minva de funcionalitat, gràcies, sobretot, a la potencialitat d'alguns dels seus operadors.

Paraules clau: *XML, W3C, XQuery, Software AG, Tamino XML Server, X-Query*.

Index

Introducció

1. Justificació del *TFC*

L'*XML* (*eXtensible Markup Language*) va ser definit el 1998 pel *W3C* (*World Wide Web Consortium*) per a representar informació de manera independent a qualsevol format. Per tal d'accedir a aquesta informació, d'extreure-la i de manipular-la, el *W3C* està treballant en la definició d'un llenguatge de consultes per a *XML* anomenat *XQuery*.

L'emmagatzemament i el tractament d'aquesta informació pot gestionar-se des d'algunes bases de dades relacionals convencionals (com ara *Oracle*, des de la versió *8i*). Tanmateix, la solució òptima consisteix en gestionar aquestes dades en el seu format nadiu, sense haver de recórrer a cap conversió de formats, mantenint intacta l'estructura dels documents.

L'empresa *Software AG* distribueix un sistema nadiu, *Tamino XML Server* (actualment en la versió 4.1.4.1), que permet, a més, treballar amb dades *XML* amb dos llenguatges de consultes alternatius. D'una banda, *Tamino* proporciona una implementació de l'especificació *XQuery* del *W3C* (*Tamino XQuery 4*), i d'una altra, ofereix un llenguatge de consultes propi anomenat *X-Query*.

Per mitjà de l'estudi previ d'aquestes tecnologies, i del plantejament i de la resolució ulteriors d'un cas pràctic, el present treball de final de carrera (*TFC*) pretén realitzar un estudi comparatiu entre el llenguatge *XQuery* del *W3C* i el llenguatge *X-Query* de *Tamino*, tot valorant les possibilitats que ofereix aquest sistema nadiu en el treball amb *XML*, en relació amb el llenguatge de consultes sobre *XML* definit pel *W3C*.

2. Objectius del *TFC*

2.1. Objectius generals

Els objectius generals del TFC consisteixen en:

- Assolir els coneixements necessaris d'*XML*.
- Comprendre un sistema de bases de dades nadiu (*Tamino XML Server*).
- Conèixer diferents llenguatges de consultes per a treballar amb *XML* (*XQuery* del *W3C* i *X-Query* de *Tamino*).
- Realitzar un cas pràctic per a consolidar tots els coneixements assolits.

2.2. Objectius específics

Per tal d'assolir els objectius generals ha calgut realitzar una sèrie d'objectius específics (entesos com a tasques), els quals han estat agrupats en Proves d'Avaluació Continuada (*PAC*), seguint el model pedagògic de la Universitat Oberta de Catalunya (*UOC*):

- PAC1: Pla de Treball.
- PAC2:
  - Estudi d'*XML*.
  - Estudi d'*XQuery 1.0* del *W3C*.
  - Estudi d'*X-Query* de *Tamino*.
- PAC3:
  - Plantejament i resolucions (en *XQuery* i en *X-Query*) del cas pràctic.
  - Estudi comparatiu de les dues resolucions del cas pràctic plantejat.
- Lliurament final:
  - Memòria.
  - Presentació virtual.

3. Requeriments de programari

El desenvolupament del *TFC* ha requerit la instal·lació del següent programari:

- *Windows 2000 Server* (amb el seu darrer *Service Pack 4*).
- *Internet Information Server*.
- *Internet Explorer 6*.
- *Tamino Starter Kit 4.1.4.1*.
- *Microsoft XML Notepad*

4. Planificació del *TFC*

4.1. Temporització

La present temporització ha estat calculada considerant que les següents dates de lliurament, segons assenyala el Pla Docent del *TFC*, són improrrogables:

- 10/03/2004 → *PAC1* (Pla de Treball)
- 13/04/2004 → *PAC2*
- 17/05/2004 → *PAC3*
- 18/06/2004 → Lliurament final (Memòria i Presentació Virtual)

4.2. Anàlisi de riscos i plans de contingència

El risc fonamental considerat a l'inici del *TFC* va consistir en el fet de no disposar, o de no disposar a temps, del programari de *Tamino* amb la llicència corresponent.

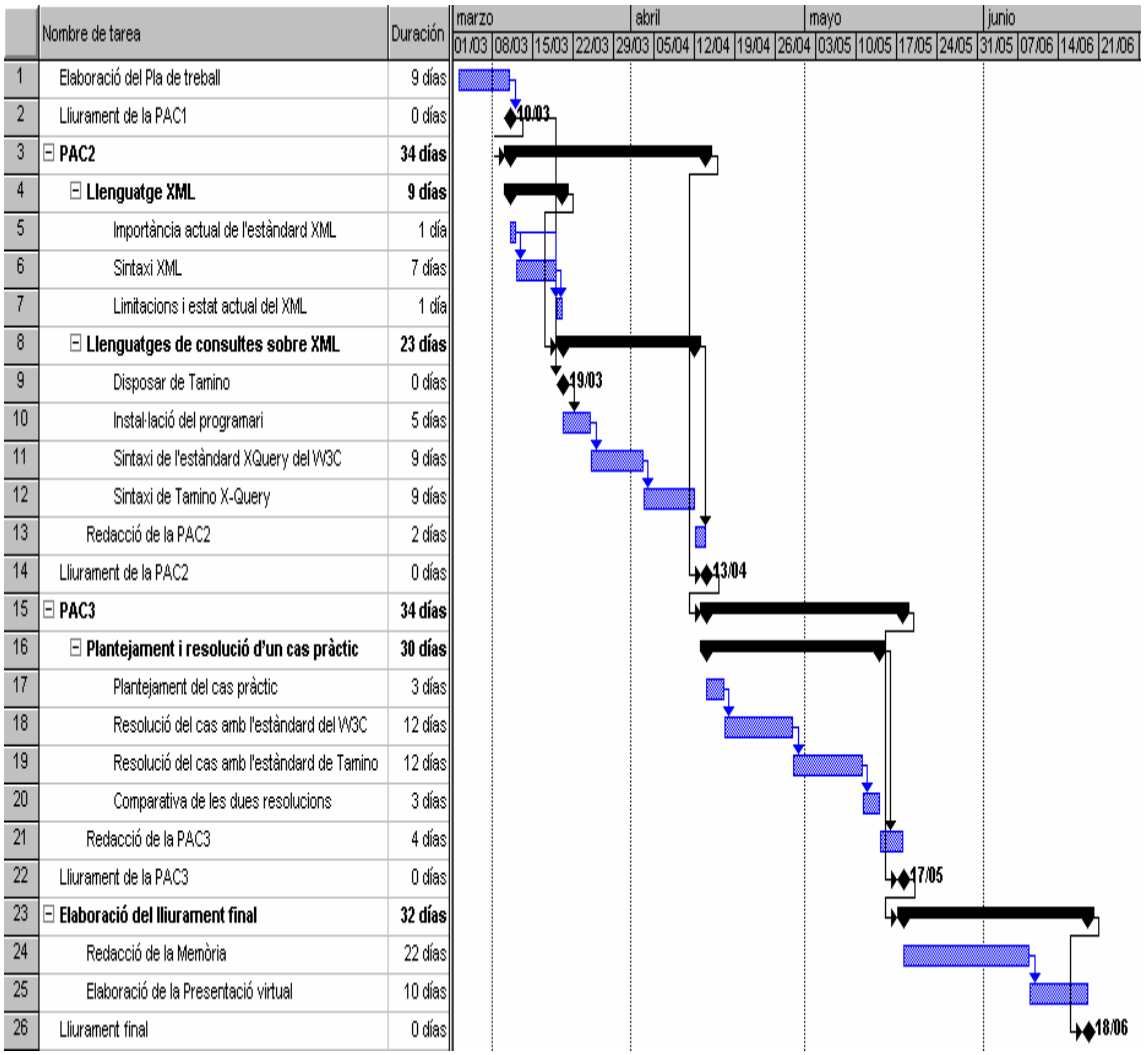
Per tal de solventar aquesta eventualitat es van considerar les següents dues possibilitats, tot i que finalment, es va optar per la primera:

- Treballar amb la *Trial Version*, gratuïta, rebent la corresponent llicència via *mail* (aquesta opció pràcticament no implicaria cap increment en el cost del TFC).
- Prescindir de *Tamino* i acudir a alguna base de dades nadiua gratuïta (en aquest cas, es pot presumir que l'increment d'hores necessàries per a trobar-ne una de prou adient, baixar-la i instal·lar-la, es pot veure compensat amb una reducció de temps proporcional a les hores necessàries per a realitzar una correcta instal·lació i ulterior ús del programari de *Tamino*).

D'altra banda, els costos en instal·lació es van calcular amb un marge de seguretat del 20%, per tal de permetre la resolució de problemes imprevistos, els quals, finalment, no es van produir.

4.3. Diagrama de Gantt

En el diagrama de Gantt que s'adjunta a continuació queden reflectides les diferents tasques i subtasques, amb els respectius costos, així com les precedències detectades entre elles i també respecte d'algunes fites. En aquest sentit, tenen especial rellevància els lliuraments de les diferents *PAC* (proves d'avaluació continuada), el lliurament final i la disponibilitat del programari de *Tamino*.



	Nombre de tarea	Duración
1	Elaboració del Pla de treball	9 días
2	Lliurament de la PAC1	0 días
3	**PAC2**	**34 días**
4	**Llenguatge XML**	**9 días**
5	Importància actual de l'estàndard XML	1 día
6	Sintaxi XML	7 días
7	Limitacions i estat actual del XML	1 día
8	**Llenguatges de consultes sobre XML**	**23 días**
9	Disposar de Tamino	0 días
10	Instal·lació del programari	5 días
11	Sintaxi de l'estàndard XQuery del W3C	9 días
12	Sintaxi de Tamino X-Query	9 días
13	Redacció de la PAC2	2 días
14	Lliurament de la PAC2	0 días
15	**PAC3**	**34 días**
16	**Plantejament i resolució d'un cas pràctic**	**30 días**
17	Plantejament del cas pràctic	3 días
18	Resolució del cas amb l'estàndard del W3C	12 días
19	Resolució del cas amb l'estàndard de Tamino	12 días
20	Comparativa de les dues resolucions	3 días
21	Redacció de la PAC3	4 días
22	Lliurament de la PAC3	0 días
23	**Elaboració del lliurament final**	**32 días**
24	Redacció de la Memòria	22 días
25	Elaboració de la Presentació virtual	10 días
26	Lliurament final	0 días

4.4. Seguiment i desviació

El *TFC*, s'ha desenvolupat al llarg de 109 díes, tenint en compte que l'enunciat de la *PAC1* va estar disponible oficialment a partir del 02/03/2004, i que el 18/06/2004 va ser la data límit per fer efectiu el lliurament final del *TFC*.

Ha estat emprat el dia natural com a unitat de quantificació de la càrrega de feina que representa cada fita. S'ha considerat que un dia equival a dues hores intenses de dedicació. Per tant, la dedicació setmanal equival a 14 hores, i la dedicació total acumulada al llarg de tot el *TFC* suma 218 hores, segons la planificació inicial.

L'esmentada planificació va presumir que 14 hores setmanals (distribuïdes lliurement al llarg de cada setmana, respectant sempre les dates improrrogables de lliurament, així com l'ordre de precedències entre tasques) havien de permetre superar amb èxit un *TFC* de 7'5 crèdits (segons consta al nou pla d'estudis d'*ETIG* a la *UOC*).

Finalment, però, s'ha constatat una desviació en el que respecta a dedicació de al voltant d'un 20% d'increment d'hores, tenint en compte la qual, s'ha emprat un total de 260 hores en la realització del *TFC*.

5. Descripció del contingut de la memòria

El capítol 1 (La tecnologia *XML*) resumeix els principals conceptes de la tecnologia *XML*. Introdueix l'estructura dels documents *XML*, els tipus de restriccions sobre les dades (especialment els esquemes) i l'especificació *XPath*, com a conceptes clau per a l'estudi posterior dels llenguatges de consultes *XQuery* i *X-Query*.

El capítol 2 (Cas pràctic sobre *Tamino XML Server*) té una doble finalitat. D'una banda, fa una introducció a l'entorn de treball de *Tamino XML Server*, eina sobre la qual s'ha dut a terme el cas pràctic que posteriorment ha permès comparar els dos llenguatges de consultes estudiats. I d'una altra, fa una presentació general del cas pràctic abans esmentat. Això permet introduir-se en el cas concret i en l'entorn de treball que, al llarg del *TFC* , s'usen tant per a descriure els diferents llenguatges, com per a fer la comparativa.

Els capítols 3 (El llenguatge de consultes *XQuery* del *W3C*) i 4 (El llenguatge de consultes *X-Query* de *Tamino*) descriuen els dos llenguatges de consultes estudiats mitjançant dos tutorials, confeccionats amb consultes executades des de *Tamino*, sobre les dades del cas pràctic del *TFC*. Ambdós tutorials introdueixen progressivament el lector en les funcionalitats respectives de cada llenguatge.

El capítol 5 (Comparativa) presenta la comparativa entre els dos llenguatges de consultes, organitzada segons es tracti de comparar operadors, funcions i altres característiques, en base a la respectiva disponibilitat, potència i simplicitat.

En el capítol 6 (Conclusions) es presenten les conclusions obtingudes després del treball de comparació dels dos llenguatges de consultes sobre *XML* analitzats.

A continuació es troba la bibliografia, on es detallen també els recursos electrònics consultats.

Finalment, l'annex A inclou els dos esquemes que defineixen l'estructura de les dades del cas pràctic ("escuderia.TSD" i "GP.TSD"), i l'annex B inclou, a tall d'exemple, dos documents *XML*, relatius a cadascun dels dos esquemes esmentats, amb dades reals ("escuderia_arrows.xml" i "GP_Alemanya.xml"). Aquests annexos poden resultar útils, d'una banda, com a exemples de sintaxi *XML*, però d'una altra, són imprescindibles per tal de conèixer l'estructura de les dades, i d'aquesta manera, poder interpretar correctament les consultes realitzades en el present treball.

Capítol 1: La tecnologia *XML*

El present capítol pretén fer un recull no exhaustiu de la versió 1.0 de l'especificació *XML* publicada l'any 1998 pel *W3C*, sintetitzant els coneixements necessaris per tal de poder treballar de manera bàsica amb documents *XML*. L'especificació oficial d'*XML* es pot trobar a l'adreça del *W3C* (*vide* referència 4 a la bibliografia).

També s'aborden altres conceptes íntimament relacionats amb l'*XML*, com l'*XML Schema* (o *XSD*) i l'*XPath*. La selecció d'aquests i d'altres aspectes relacionats amb la tecnologia *XML*, i el seu tractament més o menys a fons en aquest capítol, respon a les necessitats derivades del present *TFC*, que centra el seu interès en dos llenguatges de consultes sobre dades *XML* (*XQuery* i *X-Query*). Un estudi pormenoritzat de l'*XML* en tota la seva amplitud, i de totes les tecnologies relacionades, queda fora de l'abast d'aquest TFC.

1.1. Introducció

Als anys 70, *IBM* va inventar un llenguatge anomenat *GML* (*General Markup Language*), per a respondre a les necessitats de la mateixa empresa respecte a l'emmagatzemament i al processament de grans quantitats d'informació.

Aquest llenguatge va tenir molta acceptació entre certs àmbits de l'*ISO* (*International Organization for Standardization*), i l'any 1986 aquesta organització va crear una versió normalitzada del *GML*, anomenada *SGML* (*Standard GML*).

Com a derivacions de l'*SGML*, però orientats a Internet, van començar a aparèixer, a partir de 1989, desenvolupaments paral·lels d'un nou llenguatge d'etiquetes anomenat *HTML* (*Hiper Text Markup Language*). Des de 1996, el *W3C* treballa per a estandartzar l'*HTML*.

El mateix *W3C* va començar el 1998 a desenvolupar l'*XML*, per tal de solucionar les carències de l'*HTML* en tot allò que respecta al tractament de la informació, com ara:

- Barreja de continguts i estils en el mateix document.
- Dificultats per a compartir informació entre dispositius diferents, com per exemple entre ordinadors i telèfons mòbils.
- Presentacions en pantalla diferents, segons el visor utilitzat.

L'*XML* és un llenguatge de marques (com l'*HTML*) extensible (en el sentit que permet noves definicions de tipus de dades), que defineix un marc de treball de tipus metadades.

L'objectiu principal de l'*XML* consisteix en dotar els documents de text d'una estructura lògica, de manera que puguin ser manipulats com a dades, amb independència de la plataforma, l'arquitectura o la base de dades emprades.

1.2. Conceptes bàsics

A continuació s'ofereixen de manera esquemàtica els conceptes teòrics considerats imprescindibles per a començar a endinsar-se en la tecnologia XML:

- **Marques o etiquetes (*tags*)**

Estan formades per text que comença pel caràcter "<" i acaba pel caràcter ">", i serveixen per a delimitar l'inici i el final de cada element del document *XML*.

- **Elements**

Són els components lògics que formen els documents. Per a delimitar l'inici i el final d'un element es fan servir dues marques. Entre amdues pot haver-hi text, que

constitueix el contingut de l'element (el seu nom, i opcionalment, una sèrie d'atributs que descriuen les seves propietats):

```
<nomElement [atributs]> </nomElement>
```

Els elements buits (sense contingut) es representen amb una sola etiqueta que és al mateix temps inici i final:

```
<nomElement [atributs]/>
```

- **Atributs**

Descriuen propietats dels elements. Les cometes que encerclen el valor dels atributs poden ser simples ‘’ o dobles “” (si unes d'elles han de contenir les altres, s'ha d'alternar el tipus de cometes):

```
nomAtribut=“valorAtribut”
```

- **Instruccions de procés**

Representen comandes per als analitzadors *XML*, o bé per als programes que utilitzen el document *XML*. Han de tenir la forma següent:

```
<?instruccio atributs?>
```

1.3. Sintaxi

A continuació una llista de regles sintàctiques bàsiques per als documents *XML*:

- Les etiquetes han d'estar estructurades correctament de manera jeràrquica.
- Cada document *XML* ha de tenir un únic element arrel.
- Es distingeix entre majúscules i minúscules.
- Els espais en blanc no es tenen en compte.
- Els comentaris s'expressen de la següent manera:
  `<!-- comentari -->`
- Cal expressar uns quants caràcters reservats codificant-los amb unes marques concretes:

Codificació	Símbol
&lt;	<
&gt;	>
&amp;	&
&quot;	“
&apos;	‘

1.4. L'estructura en forma d'arbre dels documents *XML*

Els documents *XML* tenen una estructura jeràrquica en forma d'arbre amb les següents característiques:

- Es consideren nodes d'aquest arbre únicament els elements del document *XML*, però no pas els atributs dels elements, que s'inclouen dins d'aquests.
- El node pare de la resta de nodes de l'arbre es correspon amb l'element arrel del document *XML*. Aquest node arrel ha de ser únic per a tot l'arbre.
- Un node és pare d'un altre node, si l'element que representa el primer conté l'element que representa el segon.
- Els nodes germans (fills del mateix node pare) s'ordenen segons l'ordre d'aparició en el document *XML*.

1.5. Espais de noms

Quan dues o més etiquetes (tant si pertanyen al mateix document *XML*, com si estan en diferents documents *XML* als quals es fa referència des del document

analitzat) tenen el mateix nom es produeix una ambigüitat, que cal resoldre interpretant cada etiqueta de manera diferent, segons l'element en el qual es trobi.

Els espais de noms són correspondències entre un element i un *URI* (Identificador Uniforme de Recursos). Un *URI* és una cadena de caràcters que identifica un recurs a Internet independentment del protocol que es faci servir per accedir-hi. L'*URI* més comú és l'*URL* (Localitzador Uniforme de Recursos), que identifica l'adreça d'un domini a Internet fent servir el protocol *HTTP*. En el present *TFC* només s'usen *URI* de tipus *URL*.

Aquestes correspondències es fan servir per a evitar col·lisions en l'espai de noms, definint les estructures de dades que possibiliten als analitzadors sintàctics (aplicacions que comproven si els documents *XML* són sintàcticament correctes) la gestió de les esmentades col·lisions.

Els espais de noms s'utilitzen, doncs, en els documents *XML* (§ 1.6), en els esquemes *XML* (§ 1.7.2) i en els fulls d'estil *XSL* (l'*eXtensible Stylesheet Language* és un llenguatge que facilita la creació de fulls d'estil, que permeten la transformació d'una font de dades en múltiples formats de representació), per tal d'evitar col·lisions amb etiquetes amb igual nom però amb diferent significat, segons el context. Així, associant un espai de noms amb un prefix a l'element *XML*, l'analitzador *XML* podrà distingir entre dos o més espais de nom diferents, encara que el nom d'element sigui el mateix:

```
<espaiDeNoms1:nomElement [atributs]>
<espaiDeNoms2:nomElement [atributs]>
```

Prèviament al seu ús, cal declarar els espais de noms, mitjançant l'atribut `xmlns`, que té la sintaxi següent:

```
<nomElement xmlns:nomEspaiDeNoms1="uriEspaiDeNoms1"
            xmlns:nomEspaiDeNoms2="uriEspaiDeNoms2">
```

1.6. Creació de documents *XML*

Un document *XML* es divideix en dues parts: Capçalera i continguts.

1.6.1. Capçalera

La capçalera dóna informació als analitzadors i a les aplicacions *XML* de com manejar el document. La capçalera conté instruccions *XML*, les quals constitueixen un subconjunt d'instruccions de procés, amb la mateixa estructura que aquestes, que especifiquen, però, el seu nom d'element com a `xml`, i que estan destinades al mateix analitzador XML. Per exemple:

```
<?xml version="1.0" standalone="no"?>
```

Aquesta instrucció inicial indica la versió d'*XML* utilitzada (version="1.0"), i també que l'analitzador haurà de recórrer a una restricció de dades externa (`standalone="no"`), ja sigui un *DTD* (§ 1.7.1) o un esquema *XML* (§ 1.7.2), per a saber si el document XML en qüestió és vàlid.

També es podria aquí especificar, el tipus de codificació emprat. Els documents *XML* poden fer servir diferents formats de codificació dels caràcters. La codificació més adient per a documents en llengües llatines és la ISO-8859-1. Per a expressar aquesta opció, cal utilitzar la següent instrucció de procés:

```
<?xml encoding="ISO-8859-1"?>
```

Altres tipus d'instruccions poden fer referència a fulls d'estil *XSL*, passant-se la instrucció al motor *XSLT* (vol dir *XSL Transformations*, llenguatge per a la transformació de documents *XML* de tipus textual) i no a l'analitzador *XML*. Per exemple, la següent instrucció fa referència a un full d'estil per defecte, tot especificant la seva ubicació i el seu tipus:

```
<?xml-stylesheet href="fitxer.xsl" type="text/xsl"?>
```

Sovint segueix a les instruccions inicials *XML* una declaració `DOCTYPE`, amb la qual s'especifica un *DTD* (§ 1.7.1) a utilitzar en el document *XML*. Per exemple:

```
<!DOCTYPE espaiDeNoms:nomElement SYSTEM "fitxer.dtd">
```

`DOCTYPE` identifica al document *XML* actual. `SYSTEM` o `PUBLIC` (segons si l'accés és restringit o públic, respectivament) indica l'*URI* (local o no) des d'on es podrà carregar el fitxer *DTD*.

1.6.2. Contingut

El contingut del document *XML* està format per les dades pròpiament considerades (elements i atributs):

- **Element arrel**: Només pot haver-n'hi un a cada document XML, i, entre les seves etiquetes d'inici i de tancament, ha de ser-hi tota la resta de dades del document.
- **Altres elements i atributs**: La resta d'elements continguts a l'element arrel es representen per noms arbitraris que han de començar per una lletra o per un símbol de subratllat, no poden contenir espais, i diferencien entre majúscules i minúscules. Els elements poden contenir altres elements o bé atributs.

1.6.3. Documents ben formats

Per tal que un document *XML* estigui **ben format**, i que, en conseqüència, tingui sentit per a les aplicacions i els analitzadors *XML*, i per tant es pugui dur a terme alguna acció amb les dades que conté, cal que l'esmentat document respecti les següents regles:

- Tenir tancades totes les etiquetes que hagin estat obertes.
- No tenir etiquetes aniuades fora de lloc.
- Ser sintàcticament correcte d'acord amb l'especificació.

1.7. Restricció de dades *XML*: Documents vàlids

Restringir les dades *XML* consisteix en definir les combinacions vàlides dels elements i dels atributs que formen els documents *XML*.

Les restriccions de les dades són molt útils quan les aplicacions han de validar les dades amb les quals han de treballar. També poden ser útils, per exemple, quan diferents grups de persones s'han de posar d'acord sobre el format d'unes dades que s'han d'intercanviar. A més poden ajudar a d'altres desenvolupadors (i als mateixos autors, passat un cert temps) a entendre quines dades estan representades en els documents *XML*, i de quina manera.

Hi ha dos sistemes per a restringir les dades *XML*: Els *DTD* (definició de tipus de document) i els esquemes *XML*. Hom diu que un document *XML* és **vàlid** si respecta el format (una gramàtica i un conjunt concret d'etiquetes), definit en un *DTD* o en un esquema *XML*.

A continuació es descriuen tant els *DTD* com els esquemes *XML*, centrant l'estudi, però, en aquests últims, atès que en aquest *TFC* s'ha utilitzat exclusivament aquest sistema per a restringir les dades.

1.7.1. *DTD*

Els *DTD* defineixen l'estructura de les dades dels documents *XML*:
- Els elements que es permeten en un document *XML*.
- Els atributs de cada element.
- Els valors acceptables per als atributs de cada element.
- L'organització jeràrquica i les ocurrències de cada element i de qualsevol entitat externa.

Tot i formar part de l'especificació d'*XML 1.0* del *W3C*, els *DTD* no formen part, estrictament, del llenguatge *XML*.

1.7.2. Esquemes *XML*

Els esquemes són documents *XML* ben formats que serveixen (a l'igual que els *DTD*) per a processar les restriccions de les dades dels documents *XML*.

Malauradament, els esquemes *XML* no formen part de l'especificació d'*XML 1.0*. Per aquest motiu, els analitzadors del mercat, en general, només suporten part de la seva especificació, que a hores d'ara, a més, encara està incompleta.

Tot i això, l'ús dels esquemes *XML* (en lloc dels *DTD*) té molts avantatges. Per a començar, previsiblement els esquemes acabaran reemplaçant els *DTD*, en futures especificacions d'*XML*. Però a més, els esquemes *XML* suporten diferents tipus de dades, la qual cosa facilita:
- Descriure el contingut permès dels documents.
- Validar la correctesa de les dades.
- Treballar amb dades d'una base de dades.
- Definir restriccions de dades.
- Definir formats de dades.
- Convertir dades entre diferents tipus de dades.

Els esquemes *XML* han de començar amb una declaració estàndar *XML*. A continuació han de referir-se a l'espai de noms de l'esquema *XML*. En el que respecta a l'especificació de l'element arrel, a la pràctica es fa servir habitualment el terme schema.

```
<schema espaiDeNoms:"uri">
      <!- elements de l'esquema XML ->
</schema>
```

És freqüent refererir-se als esquemes *XML* com a *XSD* (*XML Schema Definition*).

- **Especificació d'elements**

Per a especificar elements es fa servir la següent sintaxi:

```
<element name="nomElement" type="tipusElement"/>
```

A `tipusElement` s'ha d'especificar un tipus de dades predefinit, o bé un de definit per l'usuari (inclou una pluralitat de predefinits). Els esquemes *XML* ofereixen un conjunt de tipus de dades molt més complet que no pas els *DTD*. A continuació s'ofereix una llista no exhaustiva d'aquests tipus de dades primitius (per a una

informació completa i actualitzada cal adreçar-se a la recomanació del *W3C* de 2 de maig de 2001 (vide referència 5 a la bibliografia).

Tipus	**Descripció**
`string`	Cadenes de caràcters
`boolean`	Vertader o fals
`float`	Coma flotant de 32 bits
`double`	Coma flotant de 64 bits
`decimal`	Decimals estàndars (positius i negatius)
`dateTime`	Data i hora
`duration`	Temps de durada
`anyURI`	Identificador Universal de Recursos

Amb el terme `complexType` podem definir a l'esquema XML tipus de dades complexes:

```
<complexType name=”nomTipus”>
      <element name=”nomElement” type=”tipusElement” />
      <element name=”nomElement” type=”tipusElement” />
      ...
</complexType>
```

Definint `nomTipus` el podrem assignar després a qualsevol element així com al seu `tipusElement`.

També existeix la possibilitat d'usar tipus de dades complexos implícits, és a dir, sense assignar-los cap nom. D'aquesta manera pot simplificar-se una mica l'esquema *XML*. Ara bé, emprar aquesta tècnica implica no poder reutilitzar aquest tipus implícit amb cap altre element.

A més, es pot fer servir un tipus implícit per a definir el contingut d'un element com a buit:

```
<element name=“nomElement”>
      <complexType content=“empty” />
</element>
```

- **Repetició d'elements**

Si un element no va acompanyat de modificadors, vol dir que ha d'aparèixer exactament un cop. Si volem, però, restringir el màxim i el mínim nombre de vegades que ha d'aparèixer un element, hem d'utilitzar els atributs `minOccurs` i `maxOccurs`:

```
<element name=”nomElement” type=”tipusElement”
minOccurs=”nombreMinimRepeticions” maxOccurs=”nombreMaximRepeticions”>
```

- **Atributs**

Els atributs es defineixen mitjançant l'element `attribute` de l'esquema *XML*:

```
<attribute name=“nomAtribut” type=“tipusAtribut” [opcionsAtribut]>
```

Els atributs disposen dels mateixos tipus de dades que els elements. Cal dir que no és obligatori que els elements tinguin atributs. La seva inclusió, d'altra banda, no ha de complicar excessivament l'aspecte de l'esquema, ja que els atributs no s'aniuen dins d'altres atributs.

Per a requerir un atribut es pot fer assignant a l'atribut `minOccurs` un valor igual a 1 (perquè, a diferència del que succeeix amb els elements, en el que respecta als atributs, el valor per defecte de `minOccurs` és 0):

```
<attribute name="nomAtribut" type="tipusAtribut  minOccurs="1" />
```

Mitjançant l'atribut `default` s'aconsegueix especificar un valor per defecte per a un atribut:

```
<attribute name="nomAtribut" type="tipusAtribut
      default="valorAtribut" />
```

Si el que volem és enumerar uns quants valors permesos per a un atribut, hem de seguir la següent notació:

```
<attribute name="nomAtribut" default="valorAtribut">
      <simpleType base="tipusSimpleDada">
            <enumeration value="valorAtribut1" />
            <enumeration value="valorAtribut2" />
      </simpleType>
</attribute>
```

- **Restriccions**

Les restriccions (també anomenades facetes) es fan servir per a controlar els valors acceptables pels elements i atributs *XML*.

Restricció	**Descripció**
`enumeration`	Defineix una llista de valors acceptables
`fractionDigits`	Especifica el nombre màxim de decimals permès (ha de ser igual o més gran que zero)
`length`	Especifica el nombre exacte de caràcters o d'elements permès (ha de ser igual o més gran que zero)
`maxExclusive`	Especifica el límit màxim pels valors numèrics (el valor ha de ser inferior a l'indicat)
`maxInclusive`	Especifica el límit màxim pels valors numèrics (el valor ha de ser inferior o igual a l'indicat)
`maxLength`	Especifica el nombre màxim de caràcters o elements permès (ha de ser igual o més gran que zero)
`minExclusive`	Especifica el límit mínim pels valors numèrics (el valor ha de ser superior a l'indicat)
`minInclusive`	Especifica el límit mínim pels valors numèrics (el valor ha de ser superior o igual a l'indicat)
`minLength`	Especifica el nombre mínim de caràcters o elements permès (ha de ser igual o més gran que zero)
`pattern`	Defineix la seqüència exacta de caràcters acceptable
`totalDigits`	Especifica el nombre exacte de dígits permès (ha de ser més gran que zero)
`whiteSpace`	Especifica com són tractats els espais en blanc (noves línies, tabulacions, espais, i retorns de carro)

1.8. L'especificació *XPath*

L'especificació *XPath* (llenguatge de camí *XML*) es troba a la recomanació del *W3C* de 16 de novembre de 1999 (*vide* referència 6 a la bibliografia). L'*XPath* defineix com localitzar un element o un atribut específic dins d'un document *XML*.

En aquesta especificació, un document *XML* és considerat com a un arbre de nodes, als quals es pot accedir especificant la seva posició dins de l'esmentat arbre. Fer referència a qualsevol node en un document *XML* pot aconseguir-se especificant la ruta relativa al node que s'estigui processant en aquest moment (node de context).

Xpath, però, també permet direccionar els nodes dels documents, de manera absoluta, respecte a l'element arrel del document (és imprescindible procedir d'aquesta manera quan s'ha de fer referència a un element que no pertany a l'àmbit del node de context).

Finalment, *XPath* defineix la sintaxi per a associar patrons, és a dir, per a trobar nodes amb un pare o un germà donats, omplint-se d'aquesta manera els espais en blanc entre les rutes absolutes i relatives.

Amb les expressions *XPath* es poden seleccionar conjunts de nodes (resultats de l'avaluació de molts, un o cap element o atribut). *XPath* també defineix, però, uns quants conjunts de funcions sobre nodes, que tenen com a paràmetre d'entrada un conjunt de nodes en forma d'expressio *XPath*, i que després retornen un resultat segons quina sigui la finalitat de cada funció en concret (`not()`, `count()`, etc.).

1.8.1. Sintaxi

L'*XPath* consisteix en un conjunt de regles sintàctiques per a delimitar parts d'un document *XML*, fent servir *paths*, de manera semblant a com es treballa amb els sistemes de fitxers dels sistemes operatius de la família *UNIX*. Una expressió *XPath* és una llista de noms d'elements fills, separats per barres ("/"), que descriu un *path* a través del document *XML*. Per exemple, la següent expressió selecciona tots els elements amb nom igual a `nomElementB` de l'element `nomElementA`:

- `/nomElementA/nomElementB`

El fet que el *path* comenci amb una barra indica que s'està fent servir un direccionament absolut a l'element `nomElementB` des de l'element arrel `nomElementA`. En canvi, si el *path* comença amb dues barres ("//"), aleshores tots els elements del document que satisfan els criteris són seleccionats, encara que estiguin a diferents nivells dins de l'arbre *XML*:

- `//nomElement`

Per a seleccionar elements desconeguts es fan servir els asteriscs ("*"):

- `//*` : Selecciona tots els elements del document.
- `/*/*/nomElement` : Selecciona tots els elements amb nom igual a `nomElement` que tinguin dos ascendents.

Per a seleccionar "branques" de l'arbre *XML* es fan servir els claudàtors ("[" i "]"):

- `/nomElementA/nomElementB[1]` : Selecciona el primer element `nomElementB` de l'element `nomElementA`.
- `/nomElementA/nomElementB[nomElementC=9.50]` : Selecciona tots els elements `nomElementB` de l'element `nomElementA` que tenen un `nomElementC` amb un valor igual a `9.50`.

Per a fer seleccions d'elements de diferents *paths*, es fa servir l'operador "|":

- `//nomElementA | //nomElementB` : Selecciona tots els elements `nomElementA` i `nomElementB` que hi hagi en el document.

En *XPath*, tots els atributs s'especifiquen amb el prefix "@":

- `//nomElement[@*]` : Selecciona tots els elements `nomElement` que tenen algun atribut.
- `//nomElement[@nomAtribut]` : Selecciona tots els elements `nomElement` que tenen algun atribut anomenat `nomAtribut`.
- `//nomElement[@nomAtribut="valorAtribut"]` : Selecciona tots els elements `nomElement` que tenen algun atribut anomenat nomAtribut amb un vaolr igual a `valorAtribut`.

1.8.2. Ubicacions: sintaxi abreujada i no abreujada

La sintaxi no abreujada per a descriure un *path* és la següent:

- `tipusNode::testNode[predicat]`

Aquestes instruccions consten, per tant, de tres elements:

- `tipusNode` : Especifica la relació entre els nodes seleccionats i el node actual (ascendents, descendents, etc.).
- `testNode` : Especifica el nom dels nodes seleccionats.
- `predicat` (opcional) : Serveix per a refinar la búsqueda dels nodes seleccionats. Els predicats es col·loquen entre dos claudàtors.

A continuació es detallen els tipus de node disponibles en la sintaxi no abreujada d'*XPath*:

Tipus d'eix	**Descripció**
`ancestor`	Conté tots els ascendents (pares, avis, etc.) del node actual, incloent el node arrel (excepte si el node actual és el mateix node arrel).
`ancestor-or-self`	Conté el node actual més tots els seus ascendents (pares, avis, etc.).
`attribute`	Conté tots els atributs del node actual.
`child`	Conté tots els fills del node actual.
`descendant`	Conté tots els descendents (fills, nets, etc.) del node actual. Aquest eix mai no conté ni atributs ni espais de noms.
`descendant-or-self`	Conté el node actual més tots els seus descendents (fills, nets, etc.).
`following`	Conté tot allò que hi hagi al document, després de l'etiqueta de tancament del node actual.
`following-sibling`	Conté tots els germans del node actual (posteriors a ell). Si el node actual és un atribut o un espai de noms, l'eix estarà buit.
`namespace`	Conté tots els espais de noms del node actual.
`parent`	Conté el pare del node actual.
`preceding`	Conté tot allò que hi hagi al document, abans de l'etiqueta d'obertura del node actual.
`preceding-sibling`	Conté tots els germans del node actual (anteriors a ell). Si el node actual és un atribut o un espai de noms, l'eix estarà buit.
`self`	Conté el node actual.

Hom pot utilitzar també una sintaxi abreujada per a descriure un *path*, la qual permet d'expressar els casos més freqüents amb molta concisió. L'abreviatura sintàctica més important consisteix en què `child::` es pot ometre. La resta d'abreviatures possibles són les següents:

Abreviatura	Significat
	child::
@	attribute::
.	self::node()
..	parent::node()
//	/descendant-or-self::node()/

1.8.3. Expressions

XPath suporta expressions numèriques, d'igualtat, relacionals i booleanes, referenciades a continuació.

- **Numèriques**

XPath sempre converteix cada operand en un nombre abans de calcular una expressió aritmètica.

Operador	Descripció
+	Suma
-	Resta
*	Multiplicació
div	Divisió
mod	Mòdul (resta de la divisió)

- **D'igualtat**

Serveixen per a comprovar la igualtat entre dos valors.

Operador	Descripció
=	Igual que
!=	No igual

- **Relacionals**

Serveixen per a comparar dos valors. *XPath* sempre converteix cada operand en un nombre abans de realitzar l'avaluació.

Operador	Descripció
<	Menor que
<=	Menor o igual que
>	Major que
>=	Major o igual que

- **Booleanes**

Serveixen per a comparar dos valors.

Operador	Descripció
or	o
and	i

1.8.4. Funcions

XPath disposa d'una llibreria de funcions de diferents tipus (d'operacions amb conjunts de nodes, amb cadenes de caràcters, numèriques i booleanes), les quals es relacionen a continuació.

- **De conjunts de nodes**

Nom	Descripció
`count()`	Retorna el nombre de nodes d'un conjunt de nodes
`id()`	Selecciona elements pel seu identificador únic
`last()`	Retorna el número de posició de l'últim node en la llista de nodes processats
`local-name()`	Retorna la part local d'un node
`name()`	Retorna el nom d'un node
`namespace-uri()`	Retorna l'*URI* de l'espai de noms del node especificat
`position()`	Retorna la posició en la llista de nodes del node que actualment està sent processat

- **D'operacions amb cadenes de caràcters**

Nom	Descripció
`concat()`	Retorna la concatenació de tots els seus arguments
`contains()`	Retorna `true` si la segona cadena està continguda dins de la primera, i `false` en cas contrari
`normalize-space()`	Treu l'espai inicial i els següents d'una cadena
`starts-with()`	Retorna true si la primera cadena comença amb la segona, i false en cas contrari
`string()`	Converteix l'argument en una cadena
`string-length()`	Retorna el nombre de caràcters d'una cadena
`substring()`	Retorna una part de la cadena que té per argument
`substring-after()`	Retorna la part de la cadena que té per argument que està a continuació de la subcadena indicada
`substring-before()`	Retorna la part de la cadena que té per argument que està abans de la subcadena indicada
`translate()`	Reemplaça un caràcter per un altre

- **Numèriques**

Nom	Descripció
`ceiling()`	Retorna l'enter més petit no inferior al nombre passat com a argument
`floor()`	Retorna l'enter més gran no superior al nombre passat com a argument
`number()`	Converteix l'argument en un nombre
`round()`	Arrodoneix un nombre a l'enter més proper
`sum()`	Retorna el valor total d'un conjunt de valors numèrics en un conjunt de nodes

- **Booleanes**

Nom	Descripció
`boolean()`	Converteix l'argument a booleà i retorna `true` o `false`
`false()`	Retorna `false`
`lang()`	Retorna `true` si l'argument del llenguatge equival a l'element `xsl:lang`, i `false` en cas contrari
`not()`	Retorna `true` si la condició passada com a argument és falsa, i `false` en cas contrari
`true()`	Retorna `true`

Capítol 2: Cas pràctic sobre *Tamino XML Server*

Per tal de fer la comparativa entre l'*XQuery* i l'*X-Query*, dins de l'entorn de *Tamino*, s'han modelitzat, a mode de cas pràctic, les dades bàsiques del Campionat del Món de Fórmula 1, corresponents a la temporada de 1998, a través d'esquemes i documents *XML*, definits posteriorment com a parts integrants d'una base de dades de *Tamino* creada *ad hoc*. Aquest ús del cas pràctic justifica la seva precedència, dins de la memòria, als apartats que tracten de l'*XQuery* i de l'*X-Query*.

2.1. Conceptes bàsics al voltant de *Tamino*

Tamino és un Sistema Gestor de Bases de Dades complet per a l'intercanvi de dades i la integració d'aplicacions basades en l'*XML,* desenvolupat per la companyia *AG Software*. Com a servidor de bases de dades nadiu *XML*, proporciona una poderosa tecnologia que optimitza l'ús d'objectes *XML*, facilita les comunicacions via Internet, emmagatzema dades *XML* i no *XML*, i possibilita l'accés a aplicacions i sistemes externs. *Tamino*, doncs, és una eina dissenyada per a emmagatzemar grans quantitats de documents *XML* en el seu format original.

És convenient conèixer els següents conceptes bàsics per a treballar des de l'entorn de *Tamino*:

- *Tamino Schema*: Els esquemes de *Tamino* segueixen un subconjunt de les directrius del *W3C* i, a més, tenen certes especificitats. Els esquemes es defineixen des d'una doble vessant:
  - Lògica (descrivint les propietats dels elements i dels atributs, així com les relacions entre tots aquests)
  - Física (descrivint com s'emmagatzemen i s'indexen físicament els documents).
- *Collection*: És un concepte específic de *Tamino*, que indica la unitat més gran d'informació possible dins d'una BD.
- *Doctype*: En general, es diu així l'element arrel d'un *DTD*. En l'àmbit de *Tamino*, en canvi, representa un contenidor d'instàncies amb el mateix element arrel, dins d'una *collection*.

2.2. Plantejament del cas pràctic

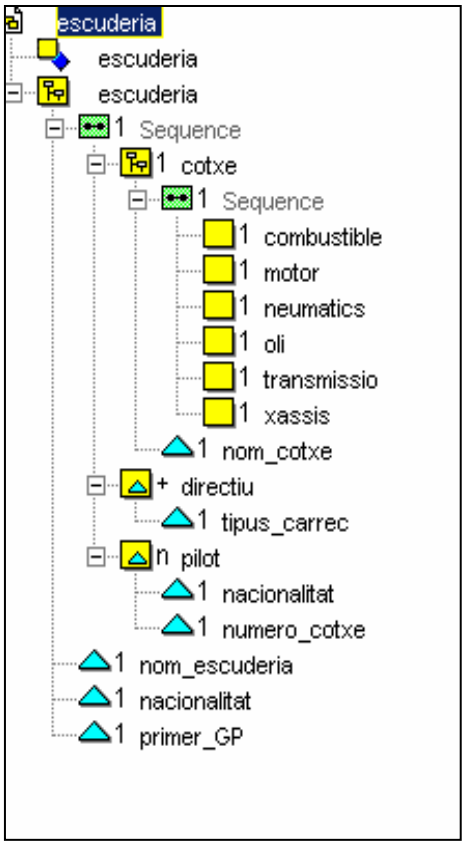
2.2.1. Modelització de les dades

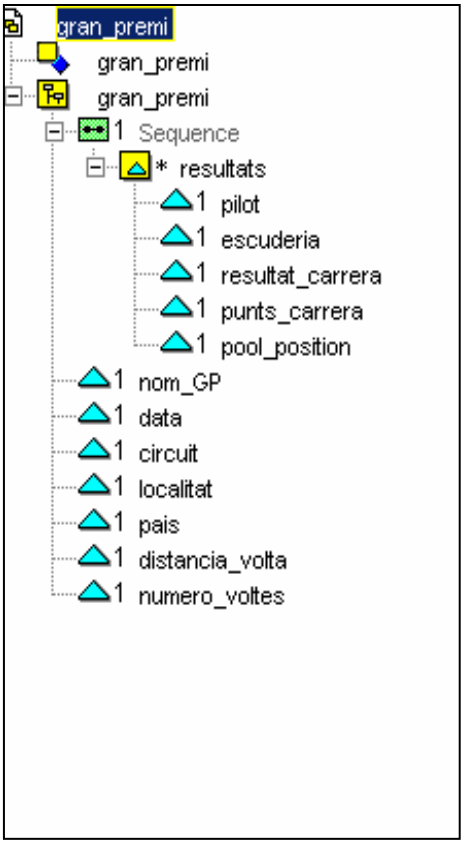
Les consultes en els dos llenguatges (*XQuery* i *X-Query*) s'executen sobre unes dades que modelitzen la temporada 1998 del Campionat del Món de Fórmula 1. L'esmentada modelització té les següents característiques:

1) La BD s'anomena "F1".
2) La *collection* de la BD s'anomena "temporada".
3) Aquesta *collection* disposa de dos *doctypes*: "escuderia" i "gran_premi".
4) A cadascun d'aquests dos *doctypes* li correspon un esquema en el format emprat per *Tamino*: "escuderia.TSD" i "GP.TSD", respectivament.
5) Onze documents *XML* contenen les dades corresponents a les onze escuderies participants en el campionat, de conformitat amb el format definit a l'esquema "escuderia.TSD". I setze documents més contenen les dades corresponents als setze grans premis en disputa, de conformitat amb el format definit a l'esquema "GP.TSD".

Per tal de poder interpretar correctament les consultes realitzades en el present treball, és necessari conèixer l'estructura de les dades. Amb aquesta finalitat s'inclouen els annexos A i B, on poden ser consultats, respectivament, els esquemes "escuderia.TSD" i "GP.TSD", així com, a tall d'exemple, dos documents *XML*, relatius a cadascun dels dos esquemes, amb dades reals.

A continuació, es mostren uns gràfics, obtinguts a partir de l'eina *Tamino Schema Editor* (§ 2.2.2) que donen idea de l'estructuració jeràrquica de les dades del cas pràctic, en forma d'arbre, tal i com estan definides en els dos esquemes abans esmentats, així com un quadre explicatiu dels diferents símbols que en ells apareixen.





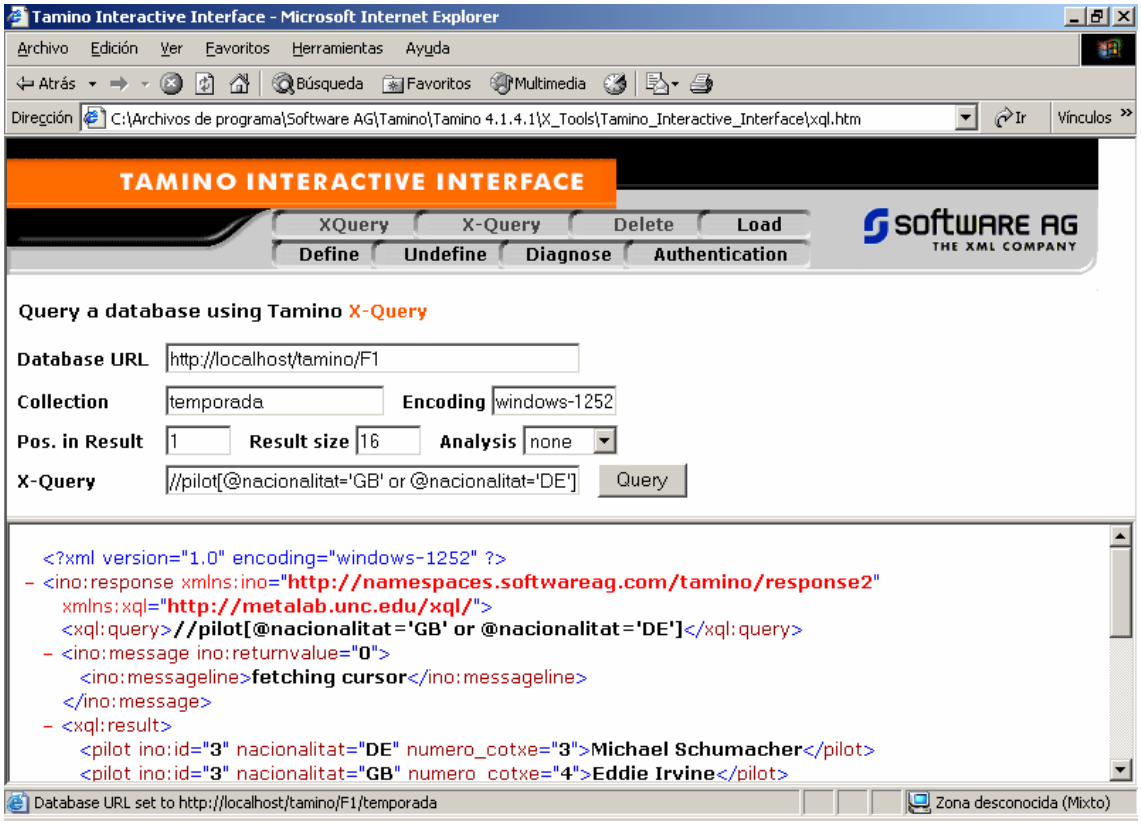
Símbol	Significat
	Esquema
	Doctype (el seu nom ha de coincidir amb el de l'element arrel)
	Seqüència d'elements en un ordre determinat
	Element complex (conté altres elements)
	Element simple
	Element amb atributs
	Atribut

2.2.2. Programari utilitzat

Durant el desenvolupament del cas pràctic s'han fet servir els següents components de *Tamino*:

- *Tamino Manager*: Eina d'administració de BD, que permet crear, esborrar, iniciar i aturar BD, definir usuaris, realitzar còpies de seguretat i restaurar-les, etc.
- *Tamino Schema Editor*: Eina visual que permet definir els esquemes, fent servir el llenguatge de definicions d'esquemes propi de *Tamino*, basat en el estàndard *XSD*, però amb certes especificitats.
- *Tamino X-Plorer*: Eina que visualitza en un arbre de navegació els continguts de *Tamino XML Server*, permetent explorar-los i manipular-los.

- *Tamino Interactive Interface*: Interfície d'usuari que permet enviar peticions a *Tamino Server* per a obtenir i guardar objectes *XML*, o parts d'ells. Des d'aquesta interfície es pot seleccionar en tot moment el llenguatge de consultes amb el qual volem treballar (*XQuery* o *X-Query*).



També s'ha fet servir la següent eina que no forma part de Tamino:
- *Microsoft XML Notepad*: Senzill i funcional editor *freeware* de documents *XML*.

2.2.3. Configuració de la *BD* i execució de consultes

Els passos seguits per a treballar amb *Tamino* han estat, breument referenciats, els següents:

1) Creació de la BD *Tamino* "F1" amb *Tamino Manager*.
2) Definició de la collection anomenada "temporada" i edició dels dos esquemes "escuderia.TSD" i "GP.TSD" amb *Tamino Schema Editor*. Seguidament, s'han hagut de definir els dos esquemes, des d'aquesta mateixa aplicació, com a part de la BD "F1" resident en el servidor de *Tamino*. Amb això els esquemes han passat a formar part de la BD de *Tamino*, restant disponibles per al seu ús posterior.
3) Edició dels documents *XML* amb *Microsoft XML Notepad*.
4) Introducció en la BD dels anteriors documents *XML*, com a instàncies dels respectius esquemes, amb *Tamino X-Plorer*. En realitzar la inserció d'aquestes dades, l'*X-Plorer* valida automàticament la seva adequació respecte als seus esquemes.
5) Obtenció dels objectes carregats a la BD, fent consultes sobre la mateixa, amb *Tamino Interactive Interface*. Cal especificar l'*URL* de la base de dades ("http://localhost/tamino/F1") i la *collection* ("temporada") amb les quals ens disposem a treballar.

Capítol 3: El llenguatge de consultes *XQuery* del *W3C*

Aquest capítol pretén sintetitzar els coneixements mínims necessaris per a treballar amb el llenguatge de consultes *XQuery*, recollits a l'esborrany de treball del *W3C* de 12 de novembre de 2003 (*vide* referència 7 a la bibliografia).

3.1. Introducció

L'especificació *XQuery* del *W3C* s'ha anat desenvolupant progressivament, per a donar resposta als requeriments identificats pel *XML Query Working Group* del *W3C*. L'objectiu d'aquest grup de treball ha consistit, en línies generals, en obtenir un llenguatge de consultes àmpliament aplicable a molts tipus diferents de fonts de dades *XML*.

L'*XQuery* deriva d'un altre llenguatge de consultes anomenat *Quilt*, el qual, a la seva vegada, va prendre característiques d'altres llenguatges, com ara l'*XPath 1.0*, l'*XQL*, l'*XML-QL*, l'*SQL* i l'*OQL*. L'*XQuery 1.0* i l'actual *XPath 2.0* tenen un origen comú, i comparteixen gran part de les respectives sintaxis.

3.2. Tutorial

A continuació es mostren diferents consultes en *XQuery*, on s'introdueixen progressivament les funcionalitats que proporciona aquest llenguatge.

Aquestes consultes han estat provades en l'entorn de *Tamino*, el qual ofereix la possibilitat de treballar amb el llenguatge *Tamino XQuery 4* (una implementació de l'estàndard *XQuery* del *W3C*). En general *Tamino* s'ha ajustat tant com ha pogut a les definicions de l'estàndard del *W3C*, tenint en compte que l'última especificació d'*XQuery* del *W3C*, de què van disposar els equips de *Software AG*, encarregats del desenvolupament de les eines necessàries per a fer possible el treball amb *XQuery* dins de l'entorn de *Tamino*, data del 15 de novembre de 2002.

Cal fer notar que, mentre s'escriuen aquestes línies, l'últim *Working Draft* del *W3C* data del 12 de novembre de 2003 (vide referència 7 a la bibliografia). *Software AG*, malgrat tot, s'ha compromès a adaptar totalment la compatibilitat del seu entorn amb l'estàndard *XQuery* del *W3C*, un cop es publiqui l'esperada *Recommendation* del *W3C*, que tindrà ja un caràcter més definitiu que no pas els actuals esborranys de treball.

Al llarg de l'exposició de les consultes seleccionades com a exemples, s'ha seguit, en general, el mateix patró, que consta dels següents elements:

- **Descripció**: Dades que es pretenen obtenir.
- **Conceptes**: Els nous conceptes que s'han introduït.
- **Consulta**: La consulta en *XQuery.*
- **Resultat**: El resultat d'executar la consulta.

Finalment, cal dir que quan en un resultat apareix un signe més "+", aquest indica que l'element a la seva dreta es mostra sense desplegar els seus nodes interiors.

XQuery 1: Funció `input()`

- **Descripció**: Obtenció de totes les dades contingudes a la BD F1.
- **Conceptes**: La funció `input()`, sense especificar res més, retorna totes les instàncies de tots els *doctypes* ("escuderia" i "gran_premi") en la *collection* indicada ("temporada").
- **Consulta**: `input()`
- **Resultat** (els tres primers):

```
+ <escuderia nacionalitat="GB" nom_escuderia="Arrows Grand Prix
International Ltd" primer_GP="Brasil 1978">
+ <escuderia nacionalitat="GB" nom_escuderia="Benetton Formula Ltd"
primer_GP="Brasil 1986">
+ <escuderia nacionalitat="IT" nom_escuderia="Ferrari SpA"
primer_GP="Monaco 1950">
```

XQuery 2: Operador inicial */*

- **Descripció**: Obtenció de tots els elements de tipus escuderia (amb tots els seus nodes descendents) en la col·lecció actual (temporada).
- **Conceptes**: L'operador "/" al principi de la consulta indica que la selecció comença al node arrel.
- **Consulta**: `input()/escuderia`
- **Resultat** (els tres primers):

```
+ <escuderia nacionalitat="GB" nom_escuderia="Arrows Grand Prix
International Ltd" primer_GP="Brasil 1978">
+ <escuderia nacionalitat="GB" nom_escuderia="Benetton Formula Ltd"
primer_GP="Brasil 1986">
+ <escuderia nacionalitat="IT" nom_escuderia="Ferrari SpA"
primer_GP="Monaco 1950">
```

XQuery 3: Operador intern */*

- **Descripció**: Obtenció de tots els elements de tipus motor que es troben en el tercer nivell de la jerarquia de nodes.
- **Conceptes**: L'operador "/" dins de l'expressió indica el següent nivell de jerarquia.
- **Consulta**: `input()/escuderia/cotxe/motor`
- **Resultat** (els tres primers):

```
<motor>TWR Arrows F1 V10 (72º)</motor>
<motor>Playlife F1-01 V10 (71º)</motor>
<motor>Ferrari 047 V10 (75º)</motor>
```

XQuery 4: Operador *//*

- **Descripció**: Obtenció de tots els elements de tipus motor, independentment dels nivells de la jerarquia de nodes que hi hagi fins a arribar a l'element desitjat.
- **Conceptes**: L'operador "//" davant d'un node indica que aquest ha de ser seleccionat independentment de la seva posició en la jerarquia de nodes. Si el nom del node no és únic, és preferible utilitzar la ruta completa fins al node desitjat, sense abreujar-la, per tal de no donar lloc a conflictes.
- **Consulta**: `input()//motor`
- **Resultat**: El mateix que a la consulta anterior.

XQuery 5: Expressions de filtrat

- **Descripció**: Obtenció de tots els motors de tipus "Ford Zetec-R V10 (72º)".
- **Conceptes**: Les expressions de filtrat es col·loquen entre claudàtors, i permeten seleccionar subconjunts de nodes. En aquestes expressions es realitzen operacions de comparació, amb diferents tipus d'operadors. Els operadors poden ser numèrics (+, -, *, div, mod), d'igualtat (=, !=), relacionals (<, <=, >, >=), i booleans (or, and).
- **Consulta**: `input()/escuderia[//motor="Ford Zetec-R V10 (72º)"]`
- **Resultat**:

```
+ <escuderia nacionalitat="IT" nom_escuderia="Minardi Team SpA"
primer_GP="Brasil 1985">
+ <escuderia nacionalitat="GB" nom_escuderia="Stewart Grand Prix"
primer_GP="Australia 1997">
```

```
+ <escuderia nacionalitat="GB" nom_escuderia="Tyrrell Racing
Organisation Ltd" primer_GP="Canada 1970">
```

XQuery 6: Selecció d'elements i d'atributs

- **Descripció**: Obtenció de tots els elements directiu que tenen un atribut tipus_carrec amb el valor "Director General".
- **Conceptes**: Les expressions contingudes entre els claudàtors filtren el conjunt de nodes indicat a la seva esquerra (els elements de tipus directiu). L'arroba (@) és l'operador atribut, i selecciona el node atribut indicat a continuació seva (els atributs de tipus tipus_carrec), adjunt al node element indicat prèviament (directiu).
- **Consulta**: `input()//directiu[@tipus_carrec="Director General"]`
- **Resultat** (els tres primers):
```
<directiu tipus_carrec="Director General">Tom Walkinshaw</directiu>
<directiu tipus_carrec="Director General">Rocco Benetton</directiu>
<directiu tipus_carrec="Director General">Luca di
Montezemolo</directiu>
```

XQuery 7: Seqüències

- **Descripció**: Obtenció dels atributs nom_escuderia i primer_GP de l'escuderia que tingui un element pilot igual a "Michael Schumacher".
- **Conceptes**: Les seqüències són concatenacions ordenades d'expressions, unides per l'operador coma (","). Les seqüències no poden contenir expressions aniuades, però sí poden contenir expressions duplicades.
- **Consulta**:
```
(input()/escuderia[pilot='Michael Schumacher']/@nom_escuderia,
input()/escuderia[pilot='Michael Schumacher']/@primer_GP)
```
- **Resultat**:
```
<xq:attribute nom_escuderia="Ferrari SpA" />
<xq:attribute primer_GP="Monaco 1950" />
```

XQuery 8: Expressions *FLWOR*

- **Descripció**: Obtenció dels atributs de tipus pilot, ordenats alfabèticament, i dels atributs de tipus resultat_carrera concordants, per al pilot que en cada gran premi ostenta un atribut pool_position igual a 1.
- **Conceptes**:
  Es fa ús d'una expressió *FLWOR*:
  - La clàusula `for` selecciona els nodes de tipus resultats dins de la variable $a.
  - La clàusula `let` és una clàusula de conveniencia que lliga la variable b a l'atribut pool_position dels elements de tipus resultats (continguts a la variable a).
  - La clàusula `where` restringeix les ocurrències de $a a aquelles que satisfan les condicions que hi ha a continuació.
  - La clàusula `return` retorna l'element elem_aux (*XQuery* permet construir nous elements, com ara elem_aux, escribint les etiquetes adients, tot combinant resultats).
  - La clàusula `sort by` (sort by de *Tamino XQuery 4* equival a `order by` de *XQuery* del *W3C*) ordena el resultat de la consulta per l'atribut pilot.
- **Consulta**:
```
for $a in input()/gran_premi/resultats
let $b := $a/@pool_position
where $b = 1
return
<elem_aux>
{ $a/@pilot }
{ $a/@resultat_carrera }
```

```
</elem_aux>
sort by (@pilot)
```

- **Resultat** (els tres primers):

```
<elem_aux pilot="David Coulthard" resultat_carrera="6" />
<elem_aux pilot="David Coulthard" resultat_carrera="15" />
<elem_aux pilot="David Coulthard" resultat_carrera="1" />
```

XQuery 9: Operacions de *join*

- **Descripció**: Obtenció dels pilots que tenen la nacionalitat corresponent al pais on es troba cada circuït, ordenat pel nom del circuït.
- **Conceptes**: Dins d'una mateixa *collection*, és possible realitzar operacions de *join*, fins i tot amb documents de diferents *doctypes*, de tal manera que el resultat de la consulta contingui nodes de tots els *doctypes* implicats per la condició de *join* implementada.
- **Consulta**:

```
for $a in input()/escuderia/pilot, $b in input()/gran_premi
where $a/@nacionalitat = $b/@pais
return
<elem_aux>
{ $b/@nom_GP }
{ $a }
</elem_aux>
sort by (@nom_GP)
```

- **Resultat** (els tres primers):

```
<elem_aux nom_GP="Gran Premi Alemanya">
  <pilot nacionalitat="DE" numero_cotxe="3">Michael
Schumacher</pilot>
</elem_aux>
<elem_aux nom_GP="Gran Premi Alemanya">
  <pilot nacionalitat="DE" numero_cotxe="10">Ralf
Schumacher</pilot>
</elem_aux>
<elem_aux nom_GP="Gran Premi Alemanya">
  <pilot nacionalitat="DE" numero_cotxe="2">Heinz Harald
Frentzen</pilot>
</elem_aux>
```

XQuery 10: Sintaxi abreujada i no abreujada. Operador comodí

- **Descripció**: Obtenció de tots atributs de l'element escuderia que té un element pilot igual a "Damon Hill".
- **Conceptes**: En *XQuery* (a l'igual que en *XPath*) una ruta pot expressar-se fent servir sintaxi abreujada o no abreujada, com en aquest cas. A continuació es pot veure la taula d'equivalències entre ambdues sintaxis:

Sintaxi abreujada	Sintaxi no abreujada	Significat
	`child::`	Fills del node actual
	`descendant::`	Descendents del node actual
`@`	`attribute::`	Atributs del node actual
`.`	`self::`	Node actual
`..`	`parent::`	Pare del node actual
`//`	`descendant-or-self::`	Node actual i tots els seus descendents

L'operador comodí (*) selecciona tots els elements.

- **Consulta**: input()/escuderia[pilot='Damon Hill']/attribute::*
- **Resultat**:

```
<xq:attribute nacionalitat="GB" />
<xq:attribute nom_escuderia="Jordan Grand Prix Ltd" />
<xq:attribute primer_GP="USA 1991" />
```

XQuery 11: Selecció de nodes segons el tipus

- **Descripció**: Obtenció de la informació textual continguda a l'element `xassis`.
- **Conceptes**: Les possibilitats per a seleccionar els nodes segons el seu tipus són les següents:

Anàlisi de node	Tipus de node seleccionat
`processing-instruction()`	Instrucció de procés
`processing-instruction('literal')`	Instrucció de procés amb nom
`comment()`	Comentari
`text()`	Text
`node()`	Qualsevol tipus
`'nom'`	Tipus principal
`'prefix:nom'`	Element o atribut en l'espai de noms indicat
`'prefix:*'`	Tots els elements i atributs en l'espai de noms indicat
`'*:nom'`	Element o atribut en tots els espais de noms possibles
`'*'`	Tots els elements i atributs

- **Consulta**:
```
for $a in input()/escuderia/cotxe/xassis/text()
return $a
```
- **Resultat** (els tres primers):
```
<xq:textNode>Arrows A19</xq:textNode>
<xq:textNode>Benetton B198</xq:textNode>
<xq:textNode>Ferrari F300</xq:textNode>
```

XQuery 12: Funció avg()

- **Descripció**: Obtenció de la distància mitja per volta de tots els circuits del campionat.
- **Conceptes**: La funció `avg()` retorna el valor mitjà dels valors numèrics de l'argument.
- **Consulta**: `avg(input()//@distancia_volta)`
- **Resultat**: `<xq:value>4.934125E3</xq:value>`

XQuery 13: Funció ceiling()

- **Descripció**: Obtenció de les distàncies totals dels grans premis.
- **Conceptes**: La funció `ceiling()` retorna l'enter més petit, que sigui igual o més gran que el valor numèric de l'argument.
- **Consulta**:
```
for $a in input()/gran_premi
return ceiling(($a/@numero_voltes * $a/@distancia_volta) div 1000)
```
- **Resultat** (els tres primers):
```
<xq:value>3.65E2</xq:value>
<xq:value>3.08E2</xq:value>
<xq:value>3.07E2</xq:value>
```

XQuery 14: Funció count()

- **Descripció**: Obtenció del nombre total d'atributs que té cada instància de l'element pilot.
- **Conceptes**: La funció `count()` retorna el nombre de nodes d'un conjunt de nodes.
- **Consulta**:
```
for $a in input()//pilot
```

```
return count($a/@*)
```
- **Resultat** (els tres primers):
```
<xq:value>2</xq:value>
<xq:value>2</xq:value>
<xq:value>2</xq:value>
```

XQuery 15: Funcions `false()` i `true()`

- **Descripció**: Obtenció del valor TRUE.
- **Conceptes**: Les funcions `false()` i `true()` retornen els valors booleans fals i cert, respectivament.
- **Consulta**: `true()`
- **Resultat**: `<xq:value>true</xq:value>`

XQuery 16: Funció `floor()`

- **Descripció**: Obtenció de la distància total dels grans premis però arrodonida a quilòmetres.
- **Conceptes**: La funció `floor()` retorna l'enter més gran, que sigui igual o més petit que el valor numèric de l'argument.
- **Consulta**:
```
for $a in input()/gran_premi
return floor(($a/@numero_voltes * $a/@distancia_volta) div 1000)
```
- **Resultat** (els tres primers):
```
<xq:value>3.64E2</xq:value>
<xq:value>3.07E2</xq:value>
<xq:value>3.06E2</xq:value>
```

XQuery 17: Funció `local-name()`

- **Descripció**: Obtenció de la part local del nom del node especificat (el primer pilot de Ferrari), sense el prefix format per l'espai de noms i els dos punts.
- **Conceptes**: La funció `local-name()` retorna el nom local d'un node.
- **Consulta**:
```
for $a in input()/escuderia
where $a/@nom_escuderia='Ferrari SpA'
return local-name($a/pilot[1])
```
- **Resultat**: `<xq:value>pilot</xq:value>`

XQuery 18: Funció `namespace-uri()`

- **Descripció**: Obtenció de l'URI de l'espai de noms del node especificat (el primer pilot de Ferrari), sense la part local del nom.
- **Conceptes**: La funció `namespace-uri()` retorna l'URI de l'espai de noms d'un node.
- **Consulta**:
```
for $a in input()/escuderia
where $a/@nom_escuderia='Ferrari SpA'
return namespace-uri($a/pilot[1])
```
- **Resultat**: Sense contingut, en aquest cas.

XQuery 19: Funció `last()`

- **Descripció**: Obtenció de l'últim pilot de l'escuderia Ferrari.
- **Conceptes**: La funció `last()` retorna el número de posició de l'últim node en la llista de nodes processats.
- **Consulta**:
```
for $a in input()/escuderia
where $a/@nom_escuderia='Ferrari SpA'
return $a/pilot[last()]
```

- **Resultat**:
  ```
  <pilot nacionalitat="GB" numero_cotxe="4">Eddie Irvine</pilot>
  ```

XQuery 20: Funcions `max()` i `min()`

- **Descripció**: Obtenció de la distància per volta més llarga de tot el campionat.
- **Conceptes**: Les funcions `max()` i `min()` retornen els valors màxim i mínim, respectivament, d'una seqüència donada.
- **Consulta**: `max(input()/gran_premi/@distancia_volta)`
- **Resultat**: `<xq:value>6968</xq:value>`

XQuery 21: Funció `normalize-space()`

- **Descripció**: Obtenció del nom de totes les escuderies sense espais en blanc innecessaris.
- **Conceptes**: La funció `normalize-space()` menysprea, si n'hi ha, els espais en blanc al principi i al final de la cadena de caràcters que rep com a paràmetre, i considerant vàlid només un espai en blanc, quan, en mig de la cadena, n'hi ha més d'un de junt.
- **Consulta**:
  ```
  for $a in input()/escuderia/@nom_escuderia
  return normalize-space($a)
  ```
- **Resultat** (els tres primers):
  ```
  <xq:value>Arrows Grand Prix International Ltd</xq:value>
  <xq:value>Benetton Formula Ltd</xq:value>
  <xq:value>Ferrari SpA</xq:value>
  ```

XQuery 22: Funció `not()`

- **Descripció**: Obtenció de la negació d'un valor booleà.
- **Conceptes**: La funció `not()` retorna `TRUE` si l'argument que rep com a paràmetre és fals, o retorna `FALSE` en cas contrari.
- **Consulta**: `not(true())`
- **Resultat**: `<xq:value>false</xq:value>`

XQuery 23: Funció `position()`

- **Descripció**: Obtenció del segon pilot de l'escuderia Ferrari.
- **Conceptes**: La funció `position()` retorna la posició en la llista de nodes del node actual.
- **Consulta**:
  ```
  for $a in input()/escuderia
  where $a/@nom_escuderia='Ferrari SpA'
  return $a/pilot[position()=2]
  ```
- **Resultat**:
  ```
  <pilot nacionalitat="GB" numero_cotxe="4">Eddie Irvine</pilot>
  ```

XQuery 24: Funció `root()`

- **Descripció**: Obtenció del node arrel per a tot element de tipus oli que sigui igual a "Elf".
- **Conceptes**: La funció `root()` retorna el node arrel del node que rep com a paràmetre.
- **Consulta**:
  ```
  for $a in input()//oli
  where $a='Elf'
  return root($a)
  ```
- **Resultat** (els tres primers):
  ```
  + <escuderia nacionalitat="GB" nom_escuderia="Arrows Grand Prix
  International Ltd" primer_GP="Brasil 1978">
  ```

```
+ <escuderia nacionalitat="GB" nom_escuderia="Jordan Grand Prix
Ltd" primer_GP="USA 1991">
+ <escuderia nacionalitat="IT" nom_escuderia="Minardi Team SpA"
primer_GP="Brasil 1985">
```

XQuery 25: Funció round()

- **Descripció**: Obtenció de les distàncies totals dels grans premis, però arrodonides a quilòmetres.
- **Conceptes**: La funció round() retorna l'enter més proper al valor numèric de l'argument.
- **Consulta**:
```
for $a in input()/gran_premi
return round(($a/@numero_voltes * $a/@distancia_volta) div 1000)
```
- **Resultat** (els tres primers):
```
<xq:value>3.65E2</xq:value>
<xq:value>3.07E2</xq:value>
<xq:value>3.06E2</xq:value>
```

XQuery 26: Funció starts-with()

- **Descripció**: Obtenció de tot aquell pilot que tingui un nom que comenci per "Jacques".
- **Conceptes**: La funció starts-with() comprova si la primera cadena de caràcters de l'argument comença per la segona, retornant TRUE en cas afirmatiu, i FALSE altrament.
- **Consulta**:
```
for $a in input()/escuderia/pilot
where starts-with($a,'Jacques')
return $a
```
- **Resultat**:
```
<pilot nacionalitat="CA" numero_cotxe="1">Jacques
Villeneuve</pilot>
```

XQuery 27: Funció string()

- **Descripció**: Obtenció de l'element pilot amb atribut numero_cotxe igual a 2 (i per tant, company d'escuderia del campió del món de F1 de la temporada anterior).
- **Conceptes**: La funció string() converteix un argument en una cadena de caràcters.
- **Consulta**:
```
for $a in input()/escuderia/pilot
where string($a/@numero_cotxe)="2"
return $a
```
- **Resultat**:
```
<pilot nacionalitat="DE" numero_cotxe="2">Heinz Harald
Frentzen</pilot>
```

XQuery 28: Funció string-join()

- **Descripció**: Obtenció del nom de l'escuderia actual del campió del món de F1 de la temporada anterior, de l'oli i del combustible que fa servir, separats per comes.
- **Conceptes**: La funció string-join() retorna una seqüència de cadenes de caràcters concatenades.
- **Consulta**:
```
for $a in input()/escuderia/pilot
where string($a/@numero_cotxe)="1"
```

```
return string-join(($a/../@nom_escuderia, $a/../cotxe/oli,
$a/../cotxe/combustible), ", ")
```

- **Resultat**:
```
<xq:value>Williams Grand Prix Engineering, Castrol,
Petrobras</xq:value>
```

XQuery 29: Funció `string-length()`

- **Descripció**: Obtenció del nombre de caràcters del nom del xassís del cotxe de cada escuderia.
- **Conceptes**: La funció `string-length()` retorna el nombre de caràcters que formen una cadena.
- **Consulta**:
```
for $a in input()//xassis
return string-length($a)
```
- **Resultat** (els tres primers):
```
<xq:value>10</xq:value>
<xq:value>13</xq:value>
<xq:value>12</xq:value>
```

XQuery 30: Funció `sum()`

- **Descripció**: Obtenció del total de voltes a circuits que els pilots han de donar per a completar la temporada.
- **Conceptes**: La funció `sum()` retorna el valor total de un conjunt de valors numèrics corresponents a un conjunt de nodes.
- **Consulta**:
```
sum(input()/gran_premi/@numero_voltes)
```
- **Resultat**: `<xq:value>1.026E3</xq:value>`

Capítol 4: El llenguatge de consultes *X-Query* de *Tamino*

Aquest capítol pretén donar una visió general del llenguatge de consultes *X-Query*, propi de *Tamino*. En primer lloc es fa una breu introducció, per a continuar després amb un tutorial que va introduïnt, progressivament, les funcionalitats d'aquest llenguatge.

4.1. Introducció

L'*X-Query* és un llenguatge de consultes sobre documents *XML* propi de *Tamino*, que es basa en l'especificació *XPath 1.0* del *W3C*, i proporciona un model de dades i una sintaxi d'expressions per a adreçar documents i parts de documents *XML*.

L'*XPath* no és un llenguatge de consultes en sí mateix, sinó més aviat un llenguatge per a referenciar parts d'un document XML.

Tamino XML Server, com a BD nadiua, permet emmagatzemar dades com a documents *XML*, i *XPath* proporciona mecanismes estàndards per a accedir als continguts dels objectes *XML* que conté una base de dades. Amb *X-Query*, però, és possible obtenir objectes *XML* fent servir mecanismes addicionals, mitjançant funcions i operadors propis.

4.2. Tutorial

A continuació es mostren diferents consultes realitzades en *X-Query*, on s'introdueixen, progressivament, les funcionalitats que proporciona aquest llenguatge.

Al llarg de l'exposició de les consultes seleccionades com a exemples, s'ha seguit, en general, el mateix patró, que consta dels següents elements:

- **Descripció**: Dades que es pretenen obtenir.
- **Conceptes**: Els nous conceptes que s'han introduït.
- **Consulta**: La consulta en *X-Query.*
- **Resultat**: El resultat d'executar la consulta.

Finalment, cal dir que quan en un resultat apareix un signe més "+", aquest indica que l'element a la seva dreta es mostra sense desplegar els seus nodes interiors.

X-Query 1: Operador inicial */*

- **Descripció**: Obtenció de tots els elements de tipus escuderia (amb tots els seus nodes descendents) en la col·lecció actual (temporada).
- **Conceptes**: L'operador "/" al principi de la consulta indica que la selecció comença al node arrel.
- **Consulta**: `/escuderia`
- **Resultat** (els tres primers):

```
+ <escuderia ino:id="1" nacionalitat="GB" nom_escuderia="Arrows
Grand Prix International Ltd" primer_GP="Brasil 1978">
+ <escuderia ino:id="2" nacionalitat="GB" nom_escuderia="Benetton
Formula Ltd" primer_GP="Brasil 1986">
+ <escuderia ino:id="3" nacionalitat="IT" nom_escuderia="Ferrari
SpA" primer_GP="Monaco 1950">
```

X-Query 2: Operador intern */*

- **Descripció**: Obtenció de tots els elements de tipus motor que es troben en el tercer nivell de la jerarquia de nodes.
- **Conceptes**: L'operador "/" dins de l'expressió indica el següent nivell de jerarquia.

- **Consulta**: `/escuderia/cotxe/motor`
- **Resultat** (els tres primers):
```
<motor ino:id="1">TWR Arrows F1 V10 (72º)</motor>
<motor ino:id="2">Playlife F1-01 V10 (71º)</motor>
<motor ino:id="3">Ferrari 047 V10 (75º)</motor>
```

X-Query 3: Operador *//*

- **Descripció**: Obtenció de tots els elements de tipus motor, independentment dels nivells de la jerarquia de nodes que hi hagi fins a arribar a l'element desitjat.
- **Conceptes**: L'operador "//" davant d'un node indica que aquest ha de ser seleccionat independentment de la seva posició en la jerarquia de nodes. Si el nom del node no és únic, és preferible utilitzar la ruta completa fins al node desitjat, sense abreujar-la, per tal de no donar lloc a conflictes.
- **Consulta**: `//motor`
- **Resultat**: idèntic a l'anterior

X-Query 4: Expressions de filtrat

- **Descripció**: Obtenció de tots els motors de tipus "Ford Zetec-R V10 (72º)".
- **Conceptes**: Les expressions de filtrat es col·loquen entre claudàtors, i permeten seleccionar subconjunts de nodes. En aquests tipus d'expressions es realitzen operacions de comparació, amb diferents tipus d'operadors. Aquests poden ser d'igualtat (`=`, `!=`), de contingut (`~=`), relacionals (`<`, `>`, `<=`, `>=`), booleans (`and`, `or`), d'ubicació (`before`, `after`, `adj`, `near`) i d'ordenació (`sortby`, `sortall`, `asc`, `desc`).
- **Consulta**: `/escuderia[//motor="Ford Zetec-R V10 (72º)"]`
- **Resultat**:
```
+ <escuderia ino:id="6" nacionalitat="IT" nom_escuderia="Minardi
Team SpA" primer_GP="Brasil 1985">
+ <escuderia ino:id="9" nacionalitat="GB" nom_escuderia="Stewart
Grand Prix" primer_GP="Australia 1997">
+ <escuderia ino:id="10" nacionalitat="GB" nom_escuderia="Tyrrell
Racing Organisation Ltd" primer_GP="Canada 1970">
```

X-Query 5: Selecció d'elements i d'atributs

- **Descripció**: Obtenció de tots els elements directiu que tenen un atribut tipus_carrec amb el valor "Director General".
- **Conceptes**: Les expressions contingudes entre els claudàtors filtren el conjunt de nodes indicat a la seva esquerra (els elements de tipus directiu). L'arroba (@) és l'operador atribut, i selecciona el node atribut indicat a continuació seva (els atributs de tipus tipus_carrec), adjunt al node element indicat prèviament (directiu).
- **Consulta**: `//directiu[@tipus_carrec="Director General"]`
- **Resultat** (els tres primers):
```
<directiu ino:id="1" tipus_carrec="Director General">Tom
Walkinshaw</directiu>
<directiu ino:id="2" tipus_carrec="Director General">Rocco
Benetton</directiu>
<directiu ino:id="3" tipus_carrec="Director General">Luca di
Montezemolo</directiu>
```

X-Query 6: Operadors booleans

- **Descripció**: Obtenció dels nodes gran_premi que o bé tenen un atribut distancia_volta amb un valor superior a 6000 metres, o bé tenen un atribut numero_voltes superior a 70.

- **Conceptes**: Els operadors booleans (and, or) poden ser usats per a expressar condicions múltiples. D'altra banda, en *X-Query* es pot obviar la barra inicial "/" que indica el node arrel.
- **Consulta**: `gran_premi[@distancia_volta>6000 or @numero_voltes>70]`
- **Resultat** (els tres primers):

```
+ <gran_premi circuit="Silverstone" data="1998-07-12"
distancia_volta="5139" ino:id="1" localitat="Northamptonshire"
nom_GP="Gran Premi Anglaterra" numero_voltes="71" pais="GB">
+ <gran_premi circuit="Hockenheim" data="1998-08-02"
distancia_volta="6823" ino:id="2" localitat="Hockenheim"
nom_GP="Gran Premi Alemanya" numero_voltes="45" pais="DE">
+ <gran_premi circuit="Oscar Alfredo Galvez" data="1998-04-12"
distancia_volta="4256" ino:id="3" localitat="Buenos Aires"
nom_GP="Gran Premi Argentina" numero_voltes="72" pais="AR">
```

X-Query 7: Operador de contingut

- **Descripció**: Obtenció dels elements de tipus cotxe que contenen la paraula "Ford".
- **Conceptes**: L'operador de contingut "~=", propi de *Tamino X-Query*, realitza una selecció basada en la cadena de caràcters situada a la seva dreta, continguda dins del node situat a la seva esquerra. El punt "." indica el node actual.
- **Consulta**: `//cotxe[.~="Ford"]`
- **Resultat**:

```
+ <cotxe ino:id="6" nom_cotxe="Minardi Ford">
+ <cotxe ino:id="9" nom_cotxe="Stewart Ford">
+ <cotxe ino:id="10" nom_cotxe="Tyrrell Ford">
```

X-Query 8: Operador comodí

- **Descripció**: Obtenció dels elements de tipus cotxe, que tenen un element descendent de tipus combustible, que comença per la lletra pe.
- **Conceptes**: L'operador de contingut "~=" pot ser usat per a realitzar seleccions amb el caràcter comodí "*", per a indicar com acaba o com comença la cadena de caràcters buscada, col·locant-lo abans o després, respectivament, del caràcter o de la subcadena indicats.
- **Consulta**: `//cotxe[combustible~="p*"]`
- **Resultat**:

```
<cotxe ino:id="8" nom_cotxe="Sauber Petronas">
 <combustible>Petronas</combustible>
 <motor>Petronas SPE01D V10 (75º)</motor>
 <neumatics>Goodyear</neumatics>
 <oli>Petronas</oli>
 <transmissio>Sauber de 6 velocitats</transmissio>
 <xassis>Sauber C17</xassis>
</cotxe>
<cotxe ino:id="11" nom_cotxe="Williams Mecachrome">
 <combustible>Petrobras</combustible>
 <motor>Mecachrome GC37/01 V10 (71º)</motor>
 <neumatics>Goodyear</neumatics>
 <oli>Castrol</oli>
 <transmissio>Williams de 6 velocitats</transmissio>
 <xassis>Williams FW20</xassis>
</cotxe>
```

X-Query 9: Operador between

- **Descripció**: Obtenció dels elements gran_premi que tenen un atribut distancia_volta amb un valor entre 5000 i 6000 metres.

- **Conceptes**: L'operador between pot ser usat per a realitzar consultes basades en un rang de valors. Les consultes amb l'operador between són més eficients que les consultes equivalents amb l'operador and.
- **Consulta**: `gran_premi[@distancia_volta between 5000,6000]`
- **Resultat** (els tres primers):

```
+ <gran_premi circuit="Silverstone" data="1998-07-12"
distancia_volta="5139" ino:id="1" localitat="Northamptonshire"
nom_GP="Gran Premi Anglaterra" numero_voltes="71" pais="GB">
+ <gran_premi circuit="Albert Park" data="1998-03-08"
distancia_volta="5303" ino:id="4" localitat="Melbourne"
nom_GP="Gran Premi Australia" numero_voltes="58" pais="AU">
+ <gran_premi circuit="Autodromo Nazionale" data="1998-09-13"
distancia_volta="5770" ino:id="12" localitat="Monza" nom_GP="Gran
Premi Italia" numero_voltes="53" pais="IT">
```

X-Query 10: Operador adj

- **Descripció**: Obtenció dels elements escuderia que tenen un atribut nom_escuderia que conté la paraula "Prix", immediatament seguida del valor "Ltd".
- **Conceptes**: L'operador adj pot ser usat per a realitzar seleccions basades en un valor adjacent a un altre.
- **Consulta**: `escuderia[@nom_escuderia~="Prix" adj "Ltd"]`
- **Resultat**:

```
+ <escuderia ino:id="4" nacionalitat="GB" nom_escuderia="Jordan
Grand Prix Ltd" primer_GP="USA 1991">
```

X-Query 11: Operador near

- **Descripció**: Obtenció dels elements escuderia que tenen un atribut nom_escuderia que conté la paraula "Prix", immediatament precedit (o seguit, si fos el cas) del valor "Grand".
- **Conceptes**: L'operador near pot ser usat per a realitzar seleccions basades en valors propers a uns altres.
- **Consulta**: `escuderia[@nom_escuderia~="Prix" near "Grand"]`
- **Resultat** (els tres primers):

```
+ <escuderia ino:id="1" nacionalitat="GB" nom_escuderia="Arrows
Grand Prix International Ltd" primer_GP="Brasil 1978">
+ <escuderia ino:id="4" nacionalitat="GB" nom_escuderia="Jordan
Grand Prix Ltd" primer_GP="USA 1991">
+ <escuderia ino:id="7" nacionalitat="FR" nom_escuderia="Prost
Grand Prix" primer_GP="Australia 1997">
```

X-Query 12: Operadors aritmètics

- **Descripció**: Obtenció dels noms dels grans premis que no superen els 300 quilòmetres de recorregut.
- **Conceptes**: A més de l'operador de multiplicació "`*`", l'*X-Query* disposa dels altres operadors aritmètics habituals en qualsevol llenguatge: `+`, `-`, `div`, `mod`. El símbol "-", com a operador unari, també pot servir per a canviar el signe d'un nombre.
- **Consulta**:

```
gran_premi[(@numero_voltes * @distancia_volta) < 300000] / @nom_GP
```

- **Resultat**:

```
<gran_premi ino:id="13" nom_GP="Gran Premi Japo" />
<gran_premi ino:id="15" nom_GP="Gran Premi Monaco" />
```

X-Query 13: Operadors d'igualtat i operador d'unió

- **Descripció**: Obtenció dels elements de tipus escuderia i de tipus gran_premi que tinguin l'atribut nacionalitat i pais (respectivament) igual a "IT".
- **Conceptes**: Les expressions amb operadors d'igualtat (=, !=) permeten comprovar si dos valors són iguals o no. L'operador "|" serveix per a seleccionar unions de conjunts de nodes.
- **Consulta**: `(/escuderia[@nacionalitat='IT'] | /gran_premi[@pais='IT'])`
- **Resultat** (els tres primers):

```
+ <escuderia ino:id="3" nacionalitat="IT" nom_escuderia="Ferrari
SpA" primer_GP="Monaco 1950">
+ <escuderia ino:id="6" nacionalitat="IT" nom_escuderia="Minardi
Team SpA" primer_GP="Brasil 1985">
+ <gran_premi circuit="Autodromo Nazionale" data="1998-09-13"
distancia_volta="5770" ino:id="12" localitat="Monza" nom_GP="Gran
Premi Italia" numero_voltes="53" pais="IT">
```

X-Query 14: Operador `intersect`

- **Descripció**: Obtenció dels elements pilot amb l'atribut nacionalitat diferent a "GB", i al mateix temps amb l'atribut numero_cotxe més gran que 4.
- **Conceptes**: L'operador `intersect` serveix per a seleccionar interseccions de conjunts de nodes.
- **Consulta**:
  `//pilot[@nacionalitat!="GB"] intersect //pilot[@numero_cotxe>4]`
  (Aquesta consulta també es podria expressar, de manera més senzilla, amb l'operador and, el qual serveix per a comprovar si les dues condicions booleanes que l'envolten són certes, retornant `TRUE` en aquest cas, i `FALSE` altrament: `//pilot[@nacionalitat!="GB" and @numero_cotxe>4]`).
- **Resultat** (els tres primers):

```
<pilot ino:id="1" nacionalitat="BR" numero_cotxe="16">Pedro
Diniz</pilot>
<pilot ino:id="1" nacionalitat="FI" numero_cotxe="17">Mika
Salo</pilot>
<pilot ino:id="2" nacionalitat="IT" numero_cotxe="5">Giancarlo
Fisichella</pilot>
```

X-Query 15: Operadors `and` i `or`

- **Descripció**: Obtenció dels elements de tipus pilot amb l'atribut nacionalitat igual a "GB" o a "DE".
- **Conceptes**: L'operador `or` comprova si alguna de les condicions booleanes que l'envolten és certa, retornant `TRUE` en aquest cas, i `FALSE` altrament. L'operador `and` comprova si totes les condicions són certes, retornant `TRUE`, o `FALSE` si no és així.
- **Consulta**: `//pilot[@nacionalitat='GB' or @nacionalitat='DE']`
- **Resultat** (els tres primers):

```
<pilot ino:id="3" nacionalitat="DE" numero_cotxe="3">Michael
Schumacher</pilot>
<pilot ino:id="3" nacionalitat="GB" numero_cotxe="4">Eddie
Irvine</pilot>
<pilot ino:id="4" nacionalitat="GB" numero_cotxe="9">Damon
Hill</pilot>
```

X-Query 16: Obtenció d'elements per posició

- **Descripció**: Obtenció del segon pilot de cada escuderia.
- **Conceptes**: Un senzill predicat entre claudàtors amb un enter, ens proporciona, d'entre un conjunt de nodes del mateix tipus, només aquell que es trobi en la posició indicada per l'enter.

- **Consulta**: `//pilot[2]`
- **Resultat** (els tres primers):
```
<pilot ino:id="1" nacionalitat="FI" numero_cotxe="17">Mika
Salo</pilot>
<pilot ino:id="2" nacionalitat="AT" numero_cotxe="6">Alexander
Wurz</pilot>
<pilot ino:id="3" nacionalitat="GB" numero_cotxe="4">Eddie
Irvine</pilot>
```

X-Query 17: Operadors before i after

- **Descripció**: Obtenció de tots els nodes descendents de l'element cotxe, que es trobin a continuació de l'element oli. Consultant l'esquema corresponent (§ A.1), es pot comprovar que només es tracta de "transmissio" i "xassis".
- **Conceptes**: Els operadors `before` i `after` serveixen per a realitzar seleccions basades en les posicions de nodes germans.
- **Consulta**: `escuderia/cotxe/* after oli`
- **Resultat** (els tres primers):
```
<transmissio ino:id="1">Arrows de 6 velocitats</transmissio>
<xassis ino:id="1">Arrows A19</xassis>
<transmissio ino:id="2">Benetton de 6 velocitats</transmissio>
<xassis ino:id="2">Benetton B198</xassis>
<transmissio ino:id="3">Ferrari de 7 velocitats</transmissio>
<xassis ino:id="3">Ferrari F300</xassis>
```

X-Query 18: Operador sortby

- **Descripció**: Obtenció dels atributs de tipus tipus_carrec igual a "Director General" en ordre alfabètic de l'element escuderia.
- **Conceptes**: Els dos punts seguits ".." indiquen l'element pare de l'element en curs. El criteri d'ordenació per defecte és ascendent, i per tant, la paraula clau asc es pot ometre.
- **Consulta**:
```
/escuderia/directiu[@tipus_carrec='Director General'] sortby(..)
```
- **Resultat** (els tres primers):
```
<directiu ino:id="1" tipus_carrec="Director General">Tom
Walkinshaw</directiu>
<directiu ino:id="2" tipus_carrec="Director General">Rocco
Benetton</directiu>
<directiu ino:id="3" tipus_carrec="Director General">Luca di
Montezemolo</directiu>
```

X-Query 19: Ordenació d'elements amb asc i desc

- **Descripció**: Obtenció dels valors de l'atribut nom_GP en ordre alfabétic.
- **Conceptes**: Amb l'operador `sortby` es pot obtenir un conjunt de nodes ordenats de certa manera. L'operador asc indica explícitament que el criteri a seguir ha de ser ascendent. L'operador desc indica que l'ordenació ha de ser descendent. El punt "." indica que el criteri d'ordenació s'ha d'aplicar al node actual.
- **Consulta**: `gran_premi/@nom_GP sortby(. asc)`
- **Resultat** (els tres primers):
```
<gran_premi ino:id="1" nom_GP="Gran Premi Alemanya" />
<gran_premi ino:id="2" nom_GP="Gran Premi Anglaterra" />
<gran_premi ino:id="3" nom_GP="Gran Premi Argentina" />
```

X-Query 20: Operador sortall

- **Descripció**: Obtenció de tots els elements de tipus pilot en ordre alfabètic descendent.

- **Conceptes**: Amb l'operador `sortall` es poden ordenar conjunts de nodes encara que pertanyin a una pluralitat de documents. L'operador desc indica que el criteri a seguir ha de ser descendent.
- **Consulta**: `/escuderia/pilot sortall(. desc)`
- **Resultat** (els tres primers):
```
<pilot ino:id="10" nacionalitat="JP" numero_cotxe="21">Toranosuke
Takagi</pilot>
<pilot ino:id="6" nacionalitat="JP" numero_cotxe="22">Shinji
Nakano</pilot>
<pilot ino:id="9" nacionalitat="BR" numero_cotxe="18">Rubens
Barrichello</pilot>
```

X-Query 21: Funció `boolean()`

- **Descripció**: Obtenció (amb quatre consultes diferents) de la taula de veritat de l'operador and.
- **Conceptes**: La funció `boolean()`, que *X-Query* comparteix amb *XPath* (a l'igual que les següents, si no s'adverteix del contrari), converteix un argument en un valor booleà (`TRUE` o `FALSE`). En les crides a funcions s'ha de fer servir, entre parèntesis, un argument per a la funció.
- **Consultes**:
```
boolean(0) and boolean(0)
boolean(0) and boolean(1)
boolean(1) and boolean(0)
boolean(1) and boolean(1)
```
- **Resultats**:
```
<xql:result>FALSE</xql:result>
<xql:result>FALSE</xql:result>
<xql:result>FALSE</xql:result>
<xql:result>TRUE</xql:result>
```

X-Query 22: Funció `number()`

- **Descripció**: Obtenció de l'element pilot amb atribut numero_cotxe igual a 1 (i per tant, campió del món de F1 de la temporada anterior).
- **Conceptes**: La funció `number()` converteix un argument en un valor numèric.
- **Consulta**: `//pilot[number(@numero_cotxe)=1]`
- **Resultat**:
```
<pilot ino:id="11" nacionalitat="CA" numero_cotxe="1">Jacques
Villeneuve</pilot>
```

X-Query 23: Funció `string()`

- **Descripció**: Obtenció de l'element pilot amb atribut numero_cotxe igual a 2 (i per tant, company d'escuderia del campió del món de F1 de la temporada anterior).
- **Conceptes**: La funció `string()` converteix un argument en una cadena de caràcters.
- **Consulta**: `//pilot[string(@numero_cotxe)="2"]`
- **Resultat**:
```
<pilot ino:id="11" nacionalitat="DE" numero_cotxe="2">Heinz Harald
Frentzen</pilot>
```

X-Query 24: Funció `ceiling()`

- **Descripció**: Obtenció de la distància total del gran premi (305474 metres), però arrodonida a quilòmetres.
- **Conceptes**: La funció `ceiling()` retorna l'enter més petit, que sigui igual o més gran que el valor numèric de l'argument.

- **Consulta**: `ceiling((number(//@numero_voltes) * number(//@distancia_volta)) div 1000)`
- **Resultat**: `<xql:result>306</xql:result>`

X-Query 25: Funció `floor()`

- **Descripció**: Obtenció de la distància total del gran premi (305474 metres), però arrodonida a quilòmetres.
- **Conceptes**: La funció `floor()` retorna l'enter més gran, que sigui igual o més petit que el valor numèric de l'argument.
- **Consulta**: `floor((number(//@numero_voltes) * number(//@distancia_volta)) div 1000)`
- **Resultat**: `<xql:result>305</xql:result>`

X-Query 26: Funció `round()`

- **Descripció**: Obtenció de la distància total del gran premi (305474 metres), però arrodonida a quilòmetres.
- **Conceptes**: La funció `round()` retorna l'enter més proper al valor numèric de l'argument.
- **Consulta**: `round((number(//@numero_voltes) * number(//@distancia_volta)) div 1000)`
- **Resultat**: `<xql:result>305</xql:result>`

X-Query 27: Funció `starts-with()`

- **Descripció**: Confirmació (amb l'obtenció d'un valor booleà) de què el nom del pilot amb número de cotxe igual a 1, comença per "Jacques".
- **Conceptes**: La funció `starts-with()` comprova si la primera cadena de caràcters de l'argument comença per la segona, retornant `TRUE` en cas afirmatiu, i `FALSE` altrament.
- **Consulta**: `starts-with(//pilot[@numero_cotxe=1],"Jacques")`
- **Resultat**: `<xql:result>TRUE</xql:result>`

X-Query 28: Funció `count()`

- **Descripció**: Obtenció del nombre total d'atributs que té definits l'element pilot (en tot el *doctype*).
- **Conceptes**: La funció `count()` retorna el nombre de nodes d'un conjunt de nodes.
- **Consulta**: `count(//pilot/@*)`
- **Resultat**: `<xql:result>44</xql:result>`

X-Query 29: Funció `sum()`

- **Descripció**: Obtenció del total de voltes a circuits que els pilots han de donar per a completar la temporada.
- **Conceptes**: La funció `sum()` retorna el valor total de un conjunt de valors numèrics corresponents a un conjunt de nodes.
- **Consulta**: `sum(gran_premi/@numero_voltes)`
- **Resultat**: `<xql:result>1026</xql:result>`

X-Query 30: Funció `name()`

- **Descripció**: Obtenció del nom de l'element fill de l'element arrel gran_premi.
- **Conceptes**: La funció `name()` retorna el nom del node indicat.
- **Consulta**: `name(/gran_premi/*)`

- **Resultat**: `<xql:result>resultats</xql:result>`

X-Query 31: Funcions `false()` i `true()`

- **Descripció**: Obtenció (amb dues consultes diferents) dels valors numèrics corresponents als valors booleans.
- **Conceptes**: Les funcions `false()` i `true()` retornen els valors booleans fals i cert, respectivament.
- **Consultes**:
```
number(false())
number(true())
```
- **Resultat**:
```
<xql:result>0</xql:result>
<xql:result>1</xql:result>
```

X-Query 32: Funció `not()`

- **Descripció**: Obtenció dels elements de tipus escuderia amb l'atribut nacionalitat diferent de "GB".
- **Conceptes**: La funció `not()` retorna cert si l'argument és fals, i retorna fals si l'argument és cert.
- **Consulta**: `/escuderia[not (@nacionalitat="GB")]`
- **Resultat** (els tres primers):
```
+ <escuderia ino:id="3" nacionalitat="IT" nom_escuderia="Ferrari
SpA" primer_GP="Monaco 1950">
+ <escuderia ino:id="6" nacionalitat="IT" nom_escuderia="Minardi
Team SpA" primer_GP="Brasil 1985">
+ <escuderia ino:id="7" nacionalitat="FR" nom_escuderia="Prost
Grand Prix" primer_GP="Australia 1997">
```

X-Query 33: Funció `last()`

- **Descripció**: Obtenció de l'últim pilot de l'escuderia Ferrari.
- **Conceptes**: La funció `last()` retorna el número de posició de l'últim node en la llista de nodes processats.
- **Consulta**: `escuderia[@nom_escuderia~="Ferrari"]/pilot[last()]`
- **Resultat**:
```
<pilot ino:id="3" nacionalitat="GB" numero_cotxe="4">Eddie
Irvine</pilot>
```

X-Query 34: Funció `position()`

- **Descripció**: Obtenció del primer pilot de l'escuderia Ferrari.
- **Conceptes**: La funció `position()` retorna el número de posició que ocupa el node de context.
- **Consulta**: `escuderia[@nom_escuderia~="Ferrari"]/pilot[position()=1]`
- **Resultat**:
```
<pilot ino:id="3" nacionalitat="DE" numero_cotxe="3">Michael
Schumacher</pilot>
```

X-Query 35: Funció `avg()`

- **Descripció**: Obtenció de la distància mitja per volta de tots els circuits del campionat.
- **Conceptes**: La funció `avg()` retorna el valor mitjà dels valors numèrics de l'argument.
- **Consulta**: `avg(//@distancia_volta)`
- **Resultat**: `<xql:result>4934.125</xql:result>`

X-Query 36: Funcions `max()` i `min()`

- **Descripció**: Obtenció de la distància per volta més petita de tots els circuits del campionat.
- **Conceptes**: La funció `min()` retorna el valor més petit dels valors numèrics de l'argument, i la funció `max()`, en canvi, retorna el valor més gran.
- **Consulta**: `min(//@distancia_volta)`
- **Resultat**: `<xql:result>3367</xql:result>`

X-Query 37: Funció `explain()`

- **Descripció**: Obtenció de dades relatives a l'execució de la consulta.
- **Conceptes**: La funció `explain()` proporciona informació al voltant de l'execució de la consulta passada com a paràmetre, per a facilitar la seva anàlisi i optimització, gràcies a dos atributs booleans (`postprocessing` i `preselection`) que indiquen la manera en què la consulta ha estat processada. En cridar aquesta funció, cal especificar l'espai de noms `ino`.
- **Consulta**: `ino:explain(escuderia[pilot ~= "Schumacher"])`
- **Resultat**:
  `<ino:explanation ino:postprocessing="TRUE" ino:preselection="FALSE" />`

Capítol 5: Comparativa

L'objectiu del present capítol és realitzar una comparativa entre els llenguatges *XQuery* (sense guió) del *W3C* i *X-Query* (amb guió) de *Tamino*, tot estudiant les possibilitats que ofereix un sistema nadiu com *Tamino*, a l'hora de treballar amb *XML*, en relació amb el llenguatge propi definit pel *W3C*.

5.1. Introducció

L'*X-Query* (amb guió) és un llenguatge de consultes sobre documents *XML*, propi de *Tamino*, que es basa en l'especificació *XPath 1.0* del *W3C*.

L'*XQuery* (sense guió) *1.0* del *W3C*, en canvi, comparteix model de dades, funcions i sintaxi amb l'especificació *XPath 2.0* del *W3C*. A més, l'*XQuery* del *W3C* també comparteix amb l'especificació *XML Schema 1.0* els mateixos tipus de dades.

Aquest diferent origen dels dos llenguatges de consultes explica per sí sol bona part de les diferències sintàctiques existents entre ambdós.

La següent taula conté les diferents propietats considerades per a realitzar la present comparativa dels dos llenguatges examinats, agrupades en tres apartats:

- Tots els operadors d'*Xquery* i d'*X-Query*.
- Totes les funcions d'*Xquery* i d'*X-Query* utilitzables dins de l'entorn de Tamino.
- Altres característiques molt específiques de cadascun dels dos llenguatges examinats.

	XQuery (W3C)	***X-Query (Tamino)***
Operadors		
Op. "/" inicial	Obligatori	Opcional
Op. d'igualtat	= !=	= !=
Op. relacionals	< > <= >=	< > <= >=
Op. lògics	and or	and or
Op. aritmètics	+ - * div mod	+ - * div mod
Op. de contingut	No	~=
Op. de rang	No	between
Op. d'ubicació	No	before after adj near
Funcions		
F. extracció dades	input() doc() collection() id() idref()	No en necessita
F. amb cadenes	starts-with() normalize-space() string-join() string-length()	starts-with()
F. numèriques	avg() max() min() sum()	avg() max() min() sum()

F. d'arrodoniment	`ceiling()`
	`floor()`
	`round()`
		`ceiling()`
		`floor()`
		`round()`

F. lògiques	`false()`
	`true()`
	`not()`
		`false()`
		`true()`
		`not()`

F. de conversió	`string()`	`string()`
		`boolean()`
		`number()`

F. amb nodes	`count()`
	`last()`
	`position()`
	`local-name()`
	`namespace-uri()`
	`processing-instruction()`
	`comment()`
	`text()`
	`node()`
	`root()`
		`count()`
		`last()`
		`position()`
		`name()`

F. anàlisi/optimització	No	`explain()`
Altres característiques		

Sintaxi abreujada	`@`
	`.`
	`..`
	`//`
		`@`
		`.`
		`..`
		`//`

Sintaxi no abreujada	`child::`
	`descendant::`
	`attribute::`
	`self::`
	`parent::`
	`descendant-or-self::`
		No

Expressions *FLWOR*	Sí	No
Constructors	`{expressio}`	No
Joining	Sí	No
Variables	`$nom_variable`	No
Filtres	`[expressio]` `where`	`[expressio]`

Ordenacions	`order by`	`sortby`
		`sortall`
		`asc`
		`desc`

Comodí	`*`	`*`
Seqüències/Unions	`expressio1, ... , expressioN`	`expressio1 | ... | expressioN`
Interseccions	`[expressio1 and expressio2]`	`[expressio1 and expressio2]` `intersect`

En els apartats següents s'expliquen les diferències i les similituds detectades entre els dos llenguatges, en totes les propietats de la taula anterior.

De cada propietat s'han avaluat tres paràmetres, exposats a continuació en l'ordre decreixent de la respectiva importància:

- **Disponibilitat**: En general, s'ha considerat com a avantatge el fet de disposar d'una funcionalitat, i com a un inconvenient el fet de no disposar-ne. En els casos en què tots dos llenguatges disposen de funcionalitats equivalents, s'ha considerat que el criteri no era determinant.
- **Potència**: Segons els resultats obtinguts amb les respectives sintaxis.
- **Simplicitat**: Segons la senzillesa de les expressions.

En cadascun dels casos examinats, o bé s'adjunten consultes d'exemple, o bé es fan referències a consultes vistes prèviament als tutorials (§3 i §4).

Finalment s'ha inclòs un apartat on es consideren les millores aconseguides en *XQuery* amb *Tamino XQuery 4*, la implementació d'*AG Software* en base a les especificacions del *W3C*, que fa possible treballar en *XQuery* dins de l'entorn de *Tamino*.

5.2. Operadors

5.2.1. Operador "/" inicial

En X-Query, la barra "/" al principi de la consulta és opcional, la qual cosa pot considerar-se un avantatge, ja que això implica una certa simplificació. Quan en X-Query s'omet aquest operador, es presumeix que el primer element de la consulta és el node arrel.

- Exemples: *XQuery* 2; *X-Query* 6.

5.2.2. Operadors d'igualtat

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 5; *X-Query* 13.

5.2.3. Operadors relacionals

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 5; *X-Query* 4.

5.2.4. Operadors lògics

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 5; *X-Query* 15.

5.2.5. Operadors aritmètics

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 5; *X-Query* 12.

5.2.6. Operador de contingut

L'operador de contingut "~=", exclusiu d'*X-Query*, realitza una selecció basada en la cadena de caràcters situada a la seva dreta, continguda dins del node situat a la seva esquerra.

Aquest potent operador comporta una simplificació notable de la sintaxi respecte a *XQuery*, ja que en aquest llenguatge, per a obtenir uns resultats molt més discrets, cal recórrer a l'us de la funció `starts-with()`.

Aquesta darrera funció (comuna a *XQuery* i a *X-Query*) permet comprovar si la primera cadena de caràcters del seu argument comença per la segona cadena. En canvi, l'operador de contingut d'*X-Query*, en combinació amb el caràcter comodí "*", permet trobar una subcadena dins d'una altra, independentment de la ubicació d'aquella dins d'aquesta.

- Exemples: *XQuery* (no disponible); *X-Query* 7.

5.2.7. Operador de rang

En *X-Query*, l'operador between pot ser usat per a realitzar consultes basades en un rang de valors. Les consultes amb l'operador between són més eficients i més senzilles que no pas les consultes equivalents (possibles tant a *XQuery* com al mateix *X-Query*) formades per dues expressions amb operadors relacionals (<, >, <=, >=), unides per l'operador and.

- Exemples: *XQuery* (no disponible); *X-Query* 9.

5.2.8. Operadors d'ubicació

X-Query, al contrari que *XQuery*, disposa d'uns quants operadors d'ubicació. Els operadors before i after poden ser usats per a realitzar seleccions basades en les posicions de nodes germans. Els operadors adj i near, per la seva banda, poden ser usats per a realitzar seleccions basades en valors adjacents o propers, respectivament, a un altre.

- Exemples: *XQuery* (no disponible); *X-Query* 10, 11 i 17.

5.3. Funcions

5.3.1. Funcions d'extracció de dades

L'*XQuery* del *W3C* utilitza diferents funcions per a extreure dades dels documents *XML*:

- doc()
- collection()
- id()
- idref()

Per exemple, doc("escuderia_williams.xml") hauria de retornar tots els nodes del document *XML* passat com a paràmetre a la funció doc().

En canvi, treballar en *X-Query* és, en aquest sentit, molt més compacte, ja que no cal emprar cap funció d'aquest tipus en les consultes. L'entorn de *Tamino* proporciona accés a totes les dades corresponents a la *collection* indicada.

5.3.2. Funcions amb cadenes

XQuery és superior a *X-Query* en el que respecta a funcions orientades al tractament de cadenes de caràcters. A més de la funció starts-with() (comuna a tots dos llenguatges), que serveix per a comprovar si la primera cadena de caràcters de l'argument comença per la segona, retornant TRUE en cas afirmatiu, i FALSE altrament, *XQuery* disposa també d'altres funcions:

- normalize-space(): Menysprea, si n'hi ha, els espais en blanc al principi i al final de la cadena de caràcters que rep com a paràmetre, i considera vàlid només un espai en blanc, quan, en mig de la cadena, n'hi ha més d'un de junt.
- string-join(): Retorna una seqüència de cadenes de caràcters concatenades amb l'operador coma ",".
- string-length(): Retorna el nombre de caràcters que formen la cadena que rep com a paràmetre.

Funció starts-with():

- Exemples: *XQuery* 26; *X-Query* 27.

Altres funcions amb cadenes:

- Exemples: *XQuery* 21, 28 i 29; *X-Query* (no disponibles).

5.3.3. Funcions numèriques

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 12, 20 i 30; *X-Query* 29, 35 i 36.

5.3.4. Funcions d'arrodoniment

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 13, 16 i 25; *X-Query* 24, 25 i 26.

5.3.5. Funcions lògiques

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 15 i 22; *X-Query* 31 i 32.

5.3.6. Funcions de conversió

X-Query supera a *XQuery* en el que respecta a funcions orientades a les conversions entre diferents tipus de dades. A més de la funció `string()` (comuna a tots dos llenguatges), que serveix per a convertir a cadena de caràcters la dada rebuda com a paràmetre, *X-Query* disposa també d'altres funcions:

- `boolean()`: Converteix un argument en un valor booleà (TRUE o FALSE).
- `number()`: Converteix un argument en un valor numèric.

Funció `string()`:

- Exemples: *XQuery* 27; *X-Query* 23.

Altres funcions de conversió:

- Exemples: *XQuery* (no disponibles); *X-Query* 21 i 22.

5.3.7. Funcions amb nodes

XQuery i *X-Query* comparteixen tres funcions de tractament de nodes:

- `count()`: Retorna el nombre de nodes d'un conjunt de nodes.
- `last()`: Retorna el número de posició de l'últim node en la llista de nodes processats.
- `position()`: Retorna la posició en la llista de nodes del node actual.

- Exemples: *XQuery* 14, 19 i 23; *X-Query* 28, 33 i 34.

Addicionalment, tant *X-Query* (amb la funció `name()`) com *XQuery* (amb la funció `local-name()`) proporcionen una altra funcionalitat comuna:

- `name()`: Retorna el nom del node indicat.
- `local-name()`: Retorna el nom local (és a dir, sense incloure el prefix de l'espai de noms) del node indicat.

- Exemples: *XQuery* 17; *X-Query* 30.

XQuery, però, proporciona, a més, moltes més funcions de tractament de nodes que no pas *X-Query*:

- `namespace-uri()`: Retorna l'*URI* de l'espai de noms d'un node.
- `processing-instruction()`: Selecciona un node de tipus *PI* (instrucció de procés).
- `comment()`: Selecciona un node de tipus comentari.
- `text()`: Selecciona un node de tipus text.
- `node()`: Selecciona un node de qualsevol tipus.
- `root()`: Retorna el node arrel del node que rep com a paràmetre.

  - Exemples: *XQuery* 18, i 24; *X-Query* (no disponibles).

5.3.8. Funcions d'anàlisi i d'optimització

X-Query disposa d'una valuosa funció que proporciona informació al voltant de l'execució de la consulta passada com a paràmetre, per a facilitar la seva anàlisi i optimització:

- `explain()`

Dos atributs booleans (`postprocessing` i `preselection`), sempre presents en el resultat de la consulta, indiquen la manera en què aquesta ha estat processada:

- L'atribut `preselection` indica si la consulta realitzarà una exploració completa del *doctype* o de la *collection* (`FALSE`) o no (`TRUE`).
- L'atribut `postprocessing` indica si el *postprocessor* (component de *Tamino* que avalua expressions amb nodes no indexats) serà cridat (`TRUE`) o no (`FALSE`).

  - Exemples: *XQuery* (no disponible); *X-Query* 37.

5.4. Altres característiques

5.4.1. Sintaxi abreujada

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

  - Exemples: *XQuery* 4; *X-Query* 3.

5.4.2. Sintaxi no abreujada

En *XQuery* una ruta pot expressar-se fent servir tant sintaxi abreujada (a l'igual que en *X-Query*) com no abreujada, la qual cosa permet donar preferència, en el plantejament de la consulta, a la seva simplificació o a la seva intel·ligibilitat, segons interessi.

  - Exemples: *XQuery* 10; *X-Query* (no disponible).

5.4.3. Expressions *FLWOR*

En *XQuery*, les expressions *FLWOR* (així dites per les clàusules que es fan servir per a construir-les: `for`, `let`, `where`, `order by` i `return`) proporcionen una potencialitat enorme en el plantejament de consultes, per complexes que aquestes puguin arribar a ser, i permeten seleccionar qualsevol dada desitjada:

- La clàusula `for` selecciona els nodes del tipus indicat dins d'una variable principal (o vàries).
- La clàusula `let` és una clàusula de conveniencia amb la qual poden introduir-se variables addicionals, i relacionar-les amb la variable principal.

- La clàusula `where` restringeix les ocurrències de la variable principal a aquelles que satisfan les condicions que hi hagi a continuació.
- La clàusula `return` indica el resultat que es vol obtenir, el qual pot consistir, fins i tot, en un nou element que combini resultats.
- La clàusula `order by` (que en *Tamino XQuery 4* ha de ser especificada com `sort by`) permet ordenar el resultat de la consulta.

    - Exemples: *XQuery* 8; *X-Query* (no disponibles).

5.4.4. Constructors

En *XQuery* (però no pas en *X-Query*) es poden construir nous elements que formaran part del document *XML* resultant de la consulta.

Un element constructor conté una expressió entre claus "{}", la qual s'avalua en executar-se la consulta. Aleshores, l'element construït és incrustat en el document *XML* de sortida.

Aquests nous elements poden ser elements complexos, amb diferents nivells jeràrquics, la qual cosa proporciona moltes possibilitats en el que respecta al format del resultat de la consulta. Els nous elements construïts d'aquesta manera poden formar part, tanmateix, d'expressions *FLWOR*.

- Exemples: *XQuery* 8; *X-Query* (no disponibles).

5.4.5. Joining

En *XQuery*, dins d'una mateixa *collection*, és possible realitzar operacions de *joininig*, fins i tot amb documents de diferents *doctypes*, de tal manera que el resultat de la consulta contingui nodes de tots els *doctypes* implicats per la condició de *join* implementada.

- Exemples: *XQuery* 9; *X-Query* (no disponible).

5.4.6. Variables

L'*XQuery* (a diferència de l'*X-Query*) admet l'ús de variables, les quals serveixen per a emmagatzemar dades, i operar posteriorment amb elles. Dins d'expressions *FLWOR*, la clàusula `for` introdueix una variable principal (o més d'una, separades per comes ",") i el valor que se li assigna. I amb la clàusula `let` poden afegir-se les variables addicionals que es considerin necessàries.

- Exemples: *XQuery* 8; *X-Query* (no disponibles).

5.4.7. Filtres

Ambdós llenguatges ofereixen les mateixes funcionalitats, però *XQuery* ofereix una sintaxi alternativa, amb la clàusula `where`, emmarcada dintre de les expressions *FLWOR*.

- Exemples: *XQuery* 5 i 8; *X-Query* 4.

5.4.8. Ordenacions

En *XQuery*, dins d'expressions *FLWOR*, es pot obtenir el resultat d'una consulta ordenat per l'element o per l'atribut indicat, fent servir la clàusula `order by` (o `sort by` en *Tamino XQuery 4*).

X-Query, però, disposa de diferents operadors d'ordenació, els quals aporten més funcionalitats:

- L'operador `sortby` serveix per a indicar a continuació el criteri d'ordenació que es vol obtenir en el resultat de la consulta.
- Amb l'operador alternatiu `sortall`, a més, es poden ordenar conjunts de nodes encara que pertanyin a una pluralitat de documents.
- En *X-Query*, el criteri d'ordenació per defecte és ascendent, i per tant, la paraula clau `asc` es pot ometre.
- Amb l'operador `desc`, en canvi, es pot indicar que el criteri a seguir ha de ser descendent.

    - Exemples: *XQuery* 8 i 9; *X-Query* 18, 19 i 20.

5.4.9. Comodí

Ambdós llenguatges ofereixen les mateixes funcionalitats i la mateixa sintaxi.

- Exemples: *XQuery* 10; *X-Query* 8.

5.4.10. Seqüències / Unions

Ambdós llenguatges ofereixen les mateixes funcionalitats tot i que amb diferent sintaxi.

- Exemples: *XQuery* 7; *X-Query* 13.

5.4.11. Interseccions

Ambdós llenguatges ofereixen les mateixes funcionalitats, però *X-Query* ofereix una sintaxi alternativa amb l'operador `intersect`.

- Exemples: *XQuery* 5; *X-Query* 14 i 15.

5.5. Resum dels avantatges identificats

La següent taula resumeix, per a diferents propietats considerades en la realització de la present comparativa, els avantatges identificats en cadascun dels dos llenguatges estudiats (*XQuery* i *X-Query*), en allò que respecta als paràmetres tinguts en compte (disponibilitat, simplicitat i potència).

	Disponibilitat	**Potència**	**Simplicitat**
Operadors:			
Op. "/" inicial			X-Query
Op. de contingut	X-Query	X-Query	X-Query
Op. de rang	X-Query	X-Query	X-Query
Op. d'ubicació	X-Query	X-Query	
Funcions:			
F. extracció dades		XQuery	X-Query
F. amb cadenes	XQuery	XQuery	
F. de conversió	X-Query	X-Query	
F. amb nodes	XQuery	XQuery	

F. anàlisi/optimització	X-Query	X-Query	
Altres característiques:			
Sintaxi no abreujada	XQuery	XQuery	
Expressions *FLWOR*	XQuery	XQuery	
Constructors	XQuery	XQuery	
Joining	XQuery	XQuery	
Variables	XQuery	XQuery	
Filtres		XQuery	
Ordenacions	X-Query	X-Query	
Interseccions	X-Query	X-Query	

S'ha considerat més important la disponibilitat que no pas la potència, i aquesta més important que la simplicitat. Quantificant la importància dels paràmetres (disponibilitat=3; potència=2; simplicitat=1) es pot establir el pes total dels avantatges detectats en *XQuery* i en *X-Query*, segons el respectiu nombre d'aparicions (ponderades) a l'anterior taula:

$$XQuery = [(7 \times 3) + (9 \times 2) + (0 \times 1)] = 39$$
$$X - Query = [(7 \times 3) + (7 \times 2) + (4 \times 1)] = 39$$

Hom pot observar com els diferents avantatges d'*XQuery* i d'*X-Query* es compensen.

5.6. Millores en *XQuery* amb *Tamino XQuery 4*

Tamino XML Server 4.1.4.1 ofereix la possibilitat de treballar, des del seu entorn, i sobre unes mateixes dades, amb *XQuery* i amb *X-Query*, indistintament. Ara bé, les consultes en *XQuery*, per tal de poder ser executades des de l'entorn de *Tamino*, han de patir petites adaptacions per a fer-les compatibles amb *Tamino XQuery 4* (la implementació de *AG Software* en base a l'especificació *XQuery* del *W3C*).

D'altra banda, *Software AG* (l'empresa desenvolupadora de *Tamino*) ha introduït certes millores en la seva definició de *Tamino XQuery 4*, respecte a l'estàndard *XQuery* del *W3C*, les quals s'exposen breument a continuació.

	***XQuery* (*W3C*)**	***Tamino Xquery 4***

Funcions d'extracció de dades	`doc()`
	`collection()`
	`id()`
	`idref()`
		`input()`

Funcions d'extracció de text	No disponibles	`containsText()`
		`containsAdjacentText()`
		`containsNearText()`
		`phonetic()`

Operadors d'actualització de documents a nivell de node	No disponibles	`update`
		`insert`
		`delete`
		`into`

5.6.1. Funcions d'extracció de dades

L'*XQuery* del *W3C* utilitza diferents funcions per a extreure dades dels documents *XML*:

- `doc()`
- `collection()`
- `id()`
- `idref()`

Per exemple, doc("escuderia_williams.xml") ha de retornar tots els nodes del document *XML* passat com a paràmetre a la funció doc().

En canvi, en *Tamino XQuery 4* (és a dir, la implementació d'*XQuery* que es fa servir dins de l'entorn de *Tamino*), només hi ha una sola funció que proporciona accés a les dades emmagatzemades en les BD *Tamino*, la qual cosa comporta una evident simplificació. Es tracta de la funció input(), que no té paràmetres, i que proporciona accés a totes les dades de la *collection* degudament indicada des de l'entorn de *Tamino*.

L'equivalent en *X-Query* a la funció input() de *Tamino XQuery 4*, de fet, és la barra inicial "/" al principi de la consulta, que de igual manera proporciona accés a totes les dades de la *collection* degudament indicada des de l'entorn de *Tamino*.

5.6.2. Funcions d'obtenció de text

Es poden realitzar operacions de búsqueda de text, amb les funcions pròpies de *Tamino XQuery 4*. Les més interessants per a la majoria d'usuaris són, posiblement, les següents:

- tf:containsText: Busca, dins del node especificat, una seqüència d'una o més paraules, i retorna TRUE si la troba, o FALSE altrament.
- tf:containsAdjacentText: Busca, dins del node especificat, una seqüència concreta d'una o més paraules, en un ordre concret, i amb una distància màxima (expressada en nombre de paraules) que pugui separar cada paraula de la seqüència de la següent. Retorna TRUE si la troba, o FALSE altrament.
- tf:containsNearText: Busca, dins del node especificat, una seqüència concreta d'una o més paraules, amb una distància màxima (expressada en nombre de paraules) que pugui separar la primera paraula de la seqüència de l'última. Retorna TRUE si la troba, o FALSE altrament.

Com que aquestes funcions pertanyen a l'espai de noms tf, diferent de l'estàndard prefixat dins de *Tamino* (xf), és necessari declarar aquest espai de noms, abans de fer qualsevol crida a les funcions que li són pròpies.

Amb la següent consulta d'exemple es vol obtenir el nom de les escuderies que contenen la subcadena de caràcters "Grand Prix":

```
declare namespace
tf="http://namespaces.softwareag.com/tamino/TaminoFunction"
for $a in input()/escuderia
where tf:containsText($a/@nom_escuderia, "Grand Prix")
return $a/@nom_escuderia
```

D'altra banda, la funció tf:phonetic() proporciona una altra facilitat (disponible de moment només en alemany), en permetre realitzar cerques sobre dades de tipus text, en base a semblances fonètiques, en cas de no conèixer la grafia exacta d'una cadena de caràcters.

5.6.3. Operacions d'actualització de documents a nivell de node

Amb unes quantes paraules clau pròpies de *Tamino XQuery 4*, es poden realitzar en els documents operacions d'inserció, modificació, canvi i eliminació de nodes o de seqüències de nodes. A continuació es mostren uns quants exemples que donen idea de l'abast d'aquestes operacions.

Amb les paraules clau update, insert i into, inserim el nou Gran Premi d'Andorra de Fórmula 1:

```
update insert
<gran_premi nom_GP="Gran Premi Andorra" data="1998-12-12" circuit="Les
valls" comu="Andorra la Vella" pais="AD" distancia_volta="500"
numero_voltes="612">
(:resta de nodes a inserir:)
</gran_premi>
into input()/gran_premi
```

Com que hem inserit "Andorra la Vella" com a atribut de tipus "comu", canviem el nom del node a "localitat", per tal d'homologar el concepte amb el dels altres grans premis:

```
update for $a in input()/gran_premi
where $a/nom_GP = "Gran Premi Andorra"
do (rename $a/@comu as localitat)
```

I finalment, amb les paraules clau update delete esborrem aquest fals gran premi:

```
update delete input()/gran_premi[nom_GP="Gran Premi Andorra"]
```

Capítol 6: Conclusions

El seguiment del present *TFC* permet arribar a certes conclusions respecte a les similituds i a les diferències sintàctiques entre els dos llenguatges de consultes estudiats (*XQuery* i *X-Query*), amb les conseqüències en funcionalitat que aquestes comporten.

D'altra banda, els coneixements adquirits durant l'execució del *TFC* permeten fer algunes previsions respecte el futur a curt i a mig termini de les tecnologies estudiades.

6.1. Avantatges d'XQuery versus avantatges d'X-Query

Després d'haver examinat exhaustivament les característiques d'*X-Query* i d'*XQuery* es pot arribar a les següents conclusions:

a) La majoria de les funcionalitats que proporcionen *XQuery* i *X-Query* són comunes als dos llenguatges, i poden obtenir-se o bé seguint el mateix procediment en ambdós casos, o bé fent ús de petites variacions sintàctiques.

b) En *XQuery*, en general, és possible d'expressar consultes més complexes que no pas en *X-Query*, les quals poden abraçar gairebé qualsevol necessitat, proporcionant una gran potència de càlcul en l'obtenció de resultats.

   Aquesta afirmació es deu, fonamentalment, a la potencialitat sintàctica de les expressions *FLWOR*, en combinació amb l'ús de variables i d'elements constructors. Aquestes expressions permeten, dins d'una mateixa *collection*, realitzar operacions de *joining*, fins i tot amb documents de diferents *doctypes*, obtenint en el resultat nodes que pertanyin a tots ells.

c) L'*X-Query*, en canvi, comporta una simplificació en la sintaxi, en relació amb l'*XQuery*, la qual cosa no sempre implica una minva de funcionalitat, gràcies sobretot a la potencialitat d'alguns dels seus operadors, i molt especialment dels següents:
   - `~=`
   - `between`
   - `desc`

Tamino XML Server ofereix la possibilitat de treballar sobre les mateixes dades, amb els dos llenguatges de consultes estudiats, segons quines siguin les característiques que li resultin més útils a l'usuari en cada moment.

6.2. Llenguatges de consultes i sistemes nadius

Finalment, cal dir que el futur prometedor (a mig termini) dels llenguatges de consultes sobre *XML* i els sistemes nadius no amaga, però, certa imatge de provisionalitat, derivada tant de les constants propostes d'innovacions, en el que respecta a les especificacions, fetes pels diferents grups de treball del *W3C* (en el que respecta a l'*XQuery*), com de la inexistència d'un estàndard *de facto* de la indústria (en el que respecta a l'*X-Query*).

Previsiblement, aquesta situació provocarà reticències durant un cert temps, encara, entre alguns directius d'empreses, a l'hora d'adoptar aquestes noves tecnologies.

Bibliografia

1) Charles F.Goldfarb, Paul Prescod, "Manual de XML". Ed. Prentice Hall, 1999.

2) Ramón Montero Ayala , "XML. Iniciación y referencia". Ed. McGraw-Hill, 2000.

Recursos electrònics

3) Recursos generals a Internet del W3C

4) "Extensible Markup Language (XML) 1.0"

5) "XML Schema Part 2: Datatypes" (recomanació del W3C de 2 de maig de 2001)

6) "XML Path Language (XPath) Version 1.0" (recomanació del W3C de 16 de novembre de 1999)

7) "XQuery 1.0: An XML Query Language"

8) Documents que descriuen i defineixen l'XQuery en la seva totalitat

9) "XQuery 1.0 and XPath 2.0 Data Model" (W3C Working Draft 12 November 2003)

10) "XQuery 1.0 and XPath 2.0 Functions and Operators" (W3C Working Draft 12 November 2003)

11) "X-Query User Guide" (documentació proporcionada per Tamino XML Server 4.1.4.1 en la distribució del seu producte)

12) "X-Query Reference Guide" (documentació proporcionada per Tamino XML Server 4.1.4.1 en la distribució del seu producte)

Annex A: Esquemes *XML* de la BD del Cas Pràctic

A.1. “escuderia.TSD”

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:tsd =
"http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition"
xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
      <tsd:schemaInfo name = "escuderia">
        <tsd:collection name = "temporada"></tsd:collection>
        <tsd:doctype name = "escuderia">
          <tsd:logical>
            <tsd:content>closed</tsd:content>
          </tsd:logical>
        </tsd:doctype>
      </tsd:schemaInfo>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name = "escuderia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name = "cotxe">
          <xs:complexType>
            <xs:sequence>
              <xs:element name = "combustible" type =
"xs:string"></xs:element>
              <xs:element name = "motor" type =
"xs:string"></xs:element>
              <xs:element name = "neumatics" type =
"xs:string"></xs:element>
              <xs:element name = "oli" type =
"xs:string"></xs:element>
              <xs:element name = "transmissio" type =
"xs:string"></xs:element>
              <xs:element name = "xassis" type =
"xs:string"></xs:element>
            </xs:sequence>
            <xs:attribute name = "nom_cotxe" type = "xs:string" use =
"required"></xs:attribute>
          </xs:complexType>
        </xs:element>
        <xs:element name = "directiu" maxOccurs = "unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base = "xs:string">
                <xs:attribute name = "tipus_carrec" type = "xs:string"
use = "required"></xs:attribute>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name = "pilot" minOccurs = "2" maxOccurs =
"unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base = "xs:string">
                <xs:attribute name = "nacionalitat" type = "xs:string"
use = "required"></xs:attribute>
```

```
                <xs:attribute name = "numero_cotxe" type = "xs:int"
use = "required"></xs:attribute>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name = "nom_escuderia" type = "xs:string" use =
"required"></xs:attribute>
      <xs:attribute name = "nacionalitat" type = "xs:string" use =
"required"></xs:attribute>
      <xs:attribute name = "primer_GP" type = "xs:string" use =
"required"></xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2. “GP.TSD”

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:tsd =
"http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition"
xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
      <tsd:schemaInfo name = "gran_premi">
        <tsd:collection name = "temporada"></tsd:collection>
        <tsd:doctype name = "gran_premi">
          <tsd:logical>
            <tsd:content>closed</tsd:content>
          </tsd:logical>
        </tsd:doctype>
      </tsd:schemaInfo>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name = "gran_premi">
    <xs:complexType>
      <xs:sequence>
        <xs:element name = "resultats" minOccurs = "0" maxOccurs =
"unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base = "xs:string">
                <xs:attribute name = "pilot" type = "xs:string" use =
"required"></xs:attribute>
                <xs:attribute name = "escuderia" type = "xs:string"
use = "required"></xs:attribute>
                <xs:attribute name = "resultat_carrera" type =
"xs:int" use = "required"></xs:attribute>
                <xs:attribute name = "punts_carrera" type = "xs:int"
use = "required"></xs:attribute>
                <xs:attribute name = "pool_position" type =
"xs:boolean" use = "required"></xs:attribute>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name = "nom_GP" type = "xs:string" use =
"required"></xs:attribute>
```

```
      <xs:attribute name = "data" type = "xs:date" use =
"required"></xs:attribute>
      <xs:attribute name = "circuit" type = "xs:string" use =
"required"></xs:attribute>
      <xs:attribute name = "localitat" type = "xs:string" use =
"required"></xs:attribute>
      <xs:attribute name = "pais" type = "xs:string" use =
"required"></xs:attribute>
      <xs:attribute name = "distancia_volta" type = "xs:int" use =
"required"></xs:attribute>
      <xs:attribute name = "numero_voltes" type = "xs:int" use =
"required"></xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Annex B: Documents XML de la BD del Cas Pràctic

B.1. “escuderia_arrows.xml”

```
<escuderia nom_escuderia="Arrows Grand Prix International Ltd"
nacionalitat="GB" primer_GP="Brasil 1978">
   <cotxe nom_cotxe="Arrows">
      <combustible>Elf</combustible>
      <motor>TWR Arrows F1 V10 (72Âº)</motor>
      <neumatics>Bridgestone</neumatics>
      <oli>Elf</oli>
      <transmissio>Arrows de 6 velocitats</transmissio>
      <xassis>Arrows A19</xassis>
   </cotxe>
   <directiu tipus_carrec="Director General">Tom Walkinshaw</directiu>
   <directiu tipus_carrec="Director Tecnic">John Barnard</directiu>
   <directiu tipus_carrec="Director Equip">John Walton</directiu>
   <directiu tipus_carrec="Cap Mecanics ">Les Jones</directiu>
   <pilot nacionalitat="BR" numero_cotxe="16">Pedro Diniz</pilot>
   <pilot nacionalitat="FI" numero_cotxe="17">Mika Salo</pilot>
</escuderia>
```

B.2. “GP_Alemanya.xml”

```
<gran_premi nom_GP="Gran Premi Alemanya" data="1998-08-02"
circuit="Hockenheim" localitat="Hockenheim" pais="DE"
distancia_volta="6823" numero_voltes="45">
   <resultats pilot="Mika Hakkinen" escuderia="McLaren International
Ltd" resultat_carrera="1" punts_carrera="10" pool_position="true"/>
   <resultats pilot="David Coulthard" escuderia="McLaren International
Ltd" resultat_carrera="2" punts_carrera="6" pool_position="false"/>
   <resultats pilot="Jacques Villeneuve" escuderia="Williams Grand
Prix Engineering" resultat_carrera="3" punts_carrera="4"
pool_position="false"/>
   <resultats pilot="Damon Hill" escuderia="Jordan Grand Prix Ltd"
resultat_carrera="4" punts_carrera="3" pool_position="false"/>
   <resultats pilot="Michael Schumacher" escuderia="Ferrari SpA"
resultat_carrera="5" punts_carrera="2" pool_position="false"/>
   <resultats pilot="Ralf Schumacher" escuderia="Jordan Grand Prix
Ltd" resultat_carrera="6" punts_carrera="1" pool_position="false"/>
</gran_premi>
```