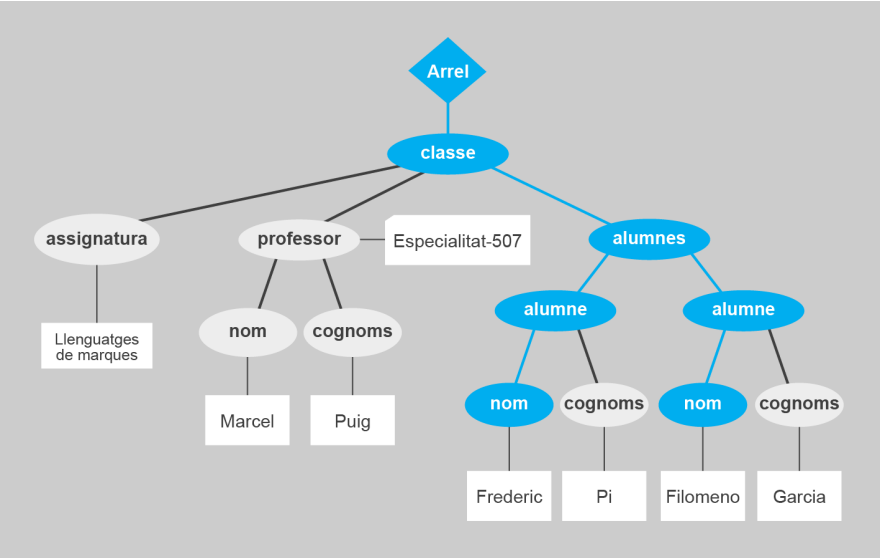
exemple, si l'expressió per avaluar fos la següent:

```
/classe/alumnes/alumne/nom
```

XPath l'avaluaria intentant aconseguir tots els camins que quadrin amb l'expressió, tal com podeu veure visualment en la figura 2.9.

Figura 2.9. Avaluació de l'expressió /classe/alumnes/alumne/nom



El resultat seran els dos resultats possibles:

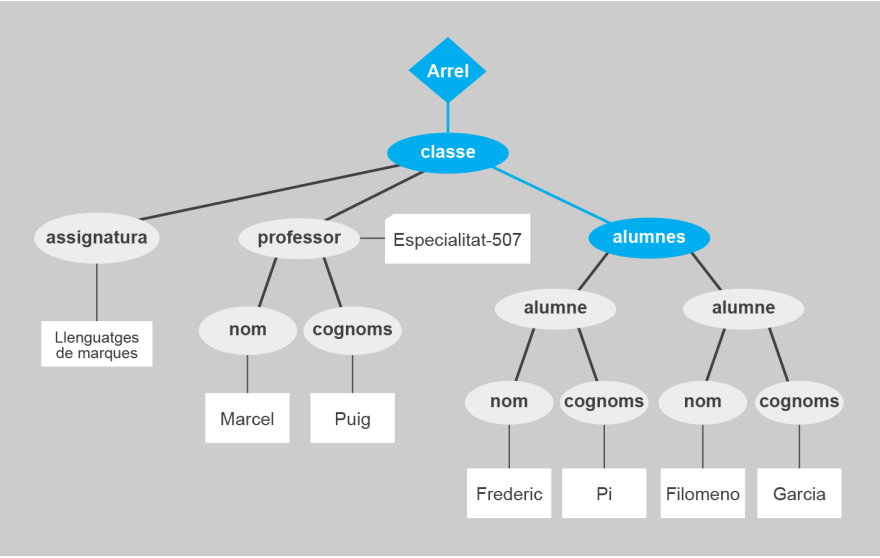
```
<nom>Frederic</nom>
<nom>Filomeno</nom>
```

A pesar que en els exemples anteriors sempre s'han retornat nodes finals això no necessàriament ha de ser així, ja que XPath pot retornar qualsevol tipus d'element com a resultat.

```
/classe/alumnes
```

La navegació per l'arbre no difereix gaire dels altres casos (figura 2.10):

Figura 2.10. Avaluació de l'expressió /classe/alumnes



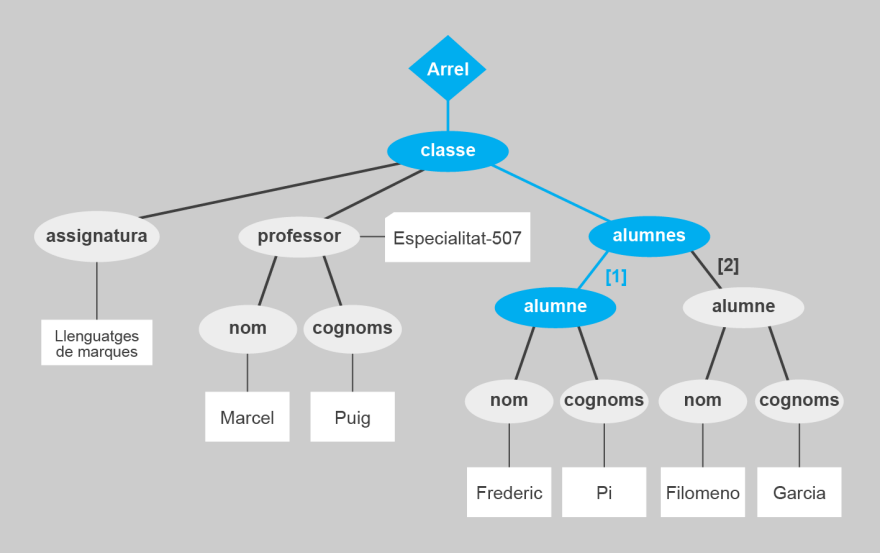
```
<alumnes>
    <alumne>
        <nom>Frederic</nom>
        <cognom>Pi</cognom>
    </alumne>
    <alumne>
        <nom>Filomeno</nom>
        <cognom>Garcia</cognom>
    </alumne>
</alumnes>
```

Si se sap que una expressió retornarà diversos resultats però només se'n vol un d'específic es pot fer servir un nombre envoltat per claudàtors quadrats "[]" per indicar quin és el que es vol aconseguir. Per retornar només el primer alumne podeu fer el següent:

```
/classe/alumnes/alumne[1]
```

Dels dos nodes disponibles com a fills de `<alumnes>`, només se seleccionarà el primer (figura 2.11).

Figura 2.11. Avaluació de l'expressió /classe/alumnes/alumne/nom



O sigui:

```
<alumne>
   <nom>Frederic</nom>
   <cognom>Pi</cognom>
</alumne>
```

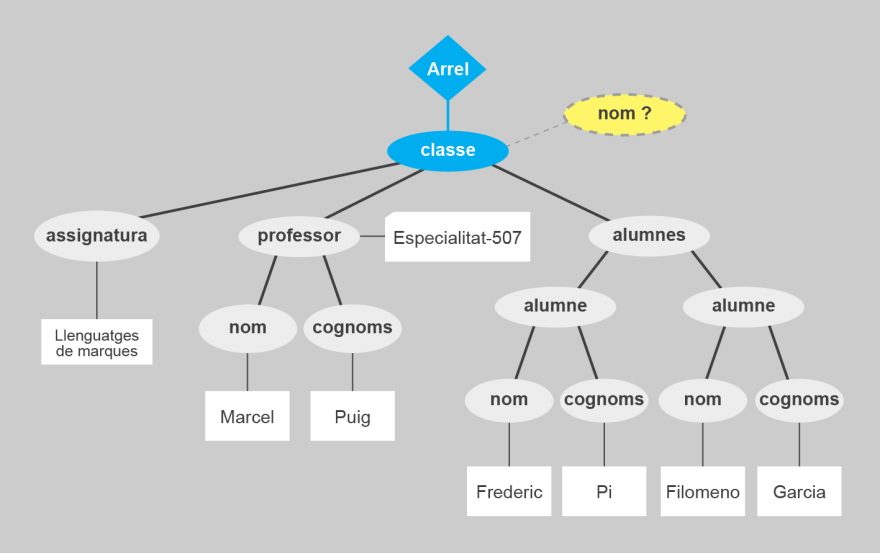
Es poden fer servir els claudàtors en qualsevol lloc de l'expressió per fer determinar quina de les branques es triarà. Per exemple, es pot obtenir només el `nom` del segon alumne amb una expressió com aquesta:

```
/classe/alumnes/alumne[2]/nom
```

Sempre s'ha d'anar en compte en escriure les expressions XPath, ja que si el camí especificat no es correspon amb un camí **real** dins de l'arbre no es retornarà cap resultat.

```
/classe/nom
```

Figura 2.12. Avaluació de l'expressió /classe/alumnes

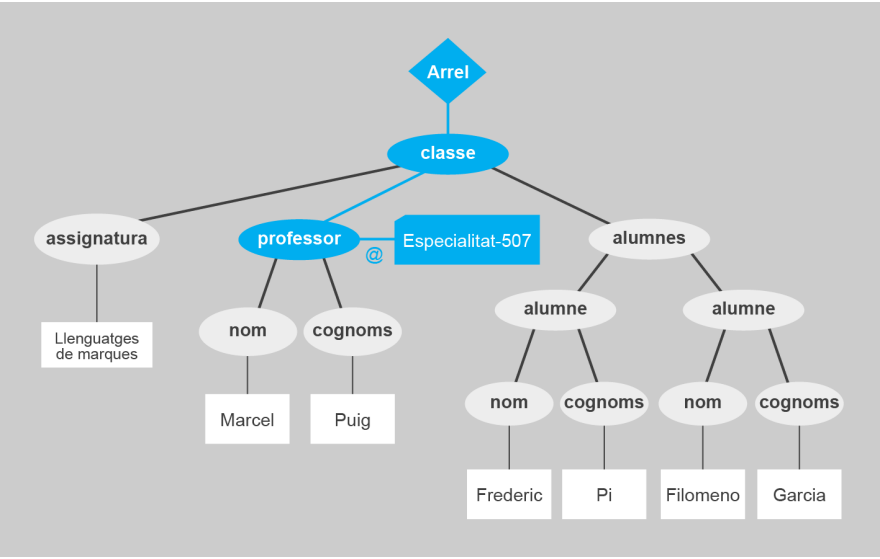


Obtenir els atributs d'un element

Els valors dels atributs es poden aconseguir especificant el símbol **@** davant del nom un cop s'hagi arribat a l'element que el conté (figura 2.13).

```
/classe/professor/@especialitat
```

Figura 2.13. Avaluació de l'expressió /classe/professor/@especialitat



S'ha de tenir en compte que a diferència del que passa amb els elements, en obtenir un atribut no tindrem un element sinó només el seu valor:

```
507
```

Obtenir el contingut d'un element

Per a aquells casos en què només vulguem el contingut de l'element, s'ha definit la funció **text()** per obtenir aquest contingut. Això s'ha fet així perquè d'altra manera, com que els nodes de text no tenen nom, no s'hi podria accedir.

De manera que si a un element que tingui contingut de dades se li afegeix *text()*:

...retornarà el contingut del node sense les etiquetes:

```
Marcel
```

Comodins

De la mateixa manera que en els sistemes operatius, es poden fer servir comodins diversos en les expressions XPath. Es poden veure els comodins en la taula 2.1.

Taula 2.1. Comodins en XPath

Comodí	Significat
*	L'asterisc es fa servir per indicar tots els elements d'un determinat nivell.
.	Com en els directoris dels sistemes operatius el punt serveix per indicar el node actual.
..	Es fa servir per indicar el pare del node en el qual estem.
/ /	Les dobles barres indiquen que quadrarà amb qualsevol cosa des del node en el qual estem. Pot ser un sol element o un arbre de nodes.

Amb l'asterisc es poden obtenir tots els elements d'un determinat nivell. Amb aquesta expressió es poden obtenir tots els elements de dins del node `professor`.

```
/classe/professor/*
```

Aquesta expressió retornarà per separat els dos nodes fills de `<professor>` (`<nom>` i `<cognom>`).

```
<nom>Marcel</nom>
<cognoms>Puig</cognoms>
```

O bé fer servir les dobles barres (//) per obtenir tots els elements `<nom>` del fitxer independentment del lloc on siguin dins del document XML.

```
//nom
```

El resultat serà:

```
<nom>Marcel</nom>
<nom>Frederic</nom>
<nom>Filomeno</nom>
```

Es poden posar les dobles barres en qualsevol lloc dins de l'expressió per indicar que hi pot haver qualsevol cosa enmig a partir del lloc on apareguin.

```
/classe/alumnes//nom
```

Donarà els dos noms dels alumnes:

```
<nom>Frederic</nom>
<nom>Filomeno</nom>
```

ser avaluades, i per tant les expressions trigaràn més a donar resultats.

Eixos XPath

Per avaluar les expressions XPath s'explora un arbre, de manera que també es proporcionen una sèrie d'elements per fer referència a parts de l'arbre. Aquests elements s'anomenen **eixos XPath** (taula 2.2).

Taula 2.2. Eixos XPath

Eix	Significat
self::	El node en el qual està el procés (fa el mateix que el punt)
child::	Fill de l'element actual
parent::	El pare de l'element actual (idèntic a fer servir `..`)
attribute::	Es fa servir per obtenir un atribut de l'element actual (@)

Alguns d'aquests eixos pràcticament no es fan servir perquè generalment és més còmode i curt definir les expressions a partir del símbol. Tothom prefereix fer servir una expressió com aquesta:

```
/classe/professor/nom
```

Que no pas la seva versió equivalent fent servir els eixos:

```
/child::classe/child::professor/child::nom
```

A part dels vistos en la taula 2.2 n'hi ha d'altres, que en aquest cas no tenen cap símbol que els simplifiqui (taula 2.3).

Taula 2.3. Eixos XPath

Eix	Significat
descendant::	Tots els descendents del node actual
desdendant-or-self::	El node actual i els seus descendents
ancestor::	Els ascendents del node
ancestor-or-self::	El node actual i els seus ascendents
prededint::	Tots els elements precedents al node actual
preceding-sibling::	Tots els germans precedents
following::	Elements que segueixen el node actual
following-sibling::	Germans posteriors al node actual
namespace::	Conté l'espai de noms del node actual

Condicions

Un apartat interessant de les expressions XPath és poder afegir condicions per a la selecció de nodes. A qualsevol expressió XPath se li poden afegir condicions per obtenir només els nodes que compleixin la condició especificada.

tors.

Per exemple, aquesta expressió selecciona només els professors que tinguin un element `<nom>` com a fill de `<professor>`:

```
/classe/professor[nom]
```

Si s'aplica l'expressió a l'exemple que hem fet servir per fer la vista d'arbre, el resultat serà el node `<professor>` que té dins seu `<nom>`.

```
<professor especialitat="507">
  <nom>Marcel</nom>
  <cognoms>Puig</cognoms>
</professor>
```

En el valor de l'expressió s'hi especifiquen camins relatius des del node que tingui la condició. Fent servir condicions es pot fer una expressió que només retorni el professor si té alumnes.

```
/classe/professor[../alumnes/alumne]
```

Normalment la complexitat de les condicions va més enllà de comprovar si el node existeix, i es fan servir per comprovar si un node té un valor determinat. Per exemple, per obtenir els professors que es diguin "Marcel":

```
/classe/professor[nom="Marcel"]
```

Les condicions es poden posar en qualsevol lloc del camí i n'hi pot haver tantes com calgui. Per obtenir el cognom del professor que es diu "Marcel" es pot fer servir una expressió com aquesta.

```
/classe/professor[nom="Marcel"]/cognoms
```

Que donarà de resultat:

```
<cognoms>Puig</cognoms>
```

De la mateixa manera que per obtenir-ne els valors, es poden fer comparacions amb els valors dels atributs especificant el seu nom rere el símbol @. Per saber si un element té l'atribut 'especialitat':

```
/classe/professor[@especialitat]
```

Retornarà:

```
<professor especialitat="507">
    <nom>Marcel</nom>
    <cognom>Puig</cognom>
</professor>
```

```
/classe/professor[@especialitat="507"]
```

```
/classe/professor[@especialitat>=507]
```

Es poden mesclar les expressions amb condicions sobre atributs i sobre elements per aconseguir expressions més complexes especificant-les una al costat de l'altra. Per exemple, podem obtenir el professor que té l'atribut especialitat a "507" i que es diu "Marcel" amb l'expressió:

```
/classe/professor[@especialitat="507"][nom="Marcel"]
```

La funció **not()** es fa servir per negar les condicions:

```
/classe/professor[not(@especialitat)]
```

L'expressió pot ser tan complexa com calgui. Per exemple es pot obtenir la llista dels cognoms dels alumnes del professor de tipus "507" que es diu "Marcel":

```
/classe/professor[@Especialitat="507"][nom="Marcel"]/../alumnes/alumne/cognoms
```

Que retornarà els dos cognoms:

```
<cognoms>Pi</cognoms>
<cognoms>Garcia</cognoms>
```

2.4.4. Seqüències

Una seqüència és una expressió XPath que retorna més d'un element. S'assemblen bastant a les llistes d'altres llenguatges:

- Tenen ordre
- Permeten duplicats
- Poden contenir valors de tipus diferent en cada terme

> La creació dinàmica de seqüències funciona a partir d'XPath 2.0.

És fàcil crear seqüències, ja que només cal tancar-les entre parèntesi i separar cada un dels termes amb comes. L'expressió següent aplicada a qualsevol document:

```
(1,2,3,4)
```

Retorna la seqüència de nombres d'un en un:

```
1
2
3
4
```

primer la primera expressió, després la segona, etc.

```
(//nom/text(), //cognoms/text())
```

Aplicat al nostre exemple retornarà primer tots els noms i després tots els cognoms:

```
Marcel
Frederic
Filomeno
Puig
Pi
Garcia
```

Unió, intersecció i disjunció

També es pot operar amb les seqüències d'elements. Una manera seria fer servir els operadors d'unió (**union**), intersecció (**intersec**) o disjunció (**except**).

Per exemple, l'expressió següent ens retornaria els cognoms dels alumnes que coincideixin amb els d'un professor:

```
(//alumne/nom) intersect (//professor/nom)
```

Amb la unió es poden unir les llistes de manera que en quedi una de sola sense duplicats:

```
(//alumne/nom) union (//professor/nom)
```

I amb la disjunció obtenim els noms de la primera seqüència que no surten en la segona:

```
(//alumne/nom) except (//professor/nom)
```

2.4.5. Funcions

> XPath ofereix una gran quantitat de funcions destinades a manipular els resultats obtinguts.

Les funcions que ofereix XPath es classifiquen en diferents grups:

- **Per manipular conjunts de nodes**: es pot obtenir el nom dels nodes, treballar amb les posicions, comptar-los, etc.
- **Per treballar amb cadenes de caràcters**: permeten extreure caràcters, concatenar, comparar... les cadenes de caràcters.
- **Per fer operacions numèriques**: es poden convertir els resultats a valors numèrics, comptar els nodes, fer-hi operacions, etc.
- **Funcions boleanes**: permeten fer operacions boleanes amb els nodes.
- **Funcions per dates**: permeten fer operacions diverses amb dates i hores.

És impossible especificar-les totes aquí, de manera que el millor és consultar l'especificació XPath (http://www.w3.org/TR/xpath-functions) per veure quines funcions es poden

Entre totes les funcions podem destacar les que surten en la taula 2.4.

Taula 2.4. Funcions XPath destacades

Funció	Ús
name()	Retorna el nom del node
sum()	Retorna la suma d'una seqüència de nodes o de valors
count()	Retorna el nombre de nodes que hi ha en una seqüència
avg()	Fa la mitjana dels valors de la seqüència
max()	Retorna el valor màxim de la seqüència
min()	Dóna el valor mínim de la seqüència
position()	Diu en quina posició es troba el node actual
last()	Retorna si el node actual és l'últim
distinct-values()	Retorna els elements de la seqüència sense duplicats
concat()	Uneix dues cadenes de caràcters
starts-with()	Retorna si la cadena comença amb els caràcters marcats
contains()	Ens diu si el resultat conté el valor
string-length()	Retorna la llargada de la cadena
substring()	Permet extreure una subcadena del resultat
string-join()	Uneix la seqüència amb el separador especificat
current-date()	Ens retornarà l'hora actual
not()	Inverteix el valors booleans

Amb les funcions es podran fer peticions que retornin valors numèrics. Per exemple, "quants alumnes tenim?":

```
count(/classe/alumnes/alumne)
```

Que tornarà el nombre d'alumnes del fitxer:

```
2
```

O bé retornar cadenes de caràcters. Per exemple, unir tots els noms separant-los per una coma:

```
string-join(//nom,",")
```

Que retornarà en un únic resultat els noms separats per una coma:

```
Marcel,Frederic,Filomeno
```

També es poden posar les funcions en els predicats. Per exemple, aquesta expressió ens retornarà els alumnes amb cognom que comenci per *p*.

```
/classe/alumnes/alumne[starts-with(cognoms,"P")]
```

En aquesta expressió volem obtenir el segon alumne de la llista:

```
/classe/alumnes/alumne[position()=2]
```