CHAPITRE2: XML & DTD: CONCEPTS DE BASE

Learning OutComes

- Quelle est la différence entre un document bien formé et un document valide?
- Quelles sont les règles syntaxique de XML?
- Qu'est ce que la DTD?

Plan

- Langage XML
  - Introduction
  - Historique
  - Champs d'application
  - Définition des structures de base
  - Règles syntaxiques de XML
- DTD

Définition de XML (eXtensible Markup Language) (1)

- Extensible
  - Avec XML on peut créer autant de balises qu'on le souhaite.
  - En HTML les balises sont prédéfinis.
- Markup
  - Balisage
  - XML permet d'identifier les différents éléments d'un document au moyen de balises.
- Language
  - C'est un langage de description de documents
  - Langage ➔ il faut respecter des règles : recommandation du W3C

Définition de XML (2)

- XML possède deux types de documents :
  - Les documents bien formés
    - Ils obéissent aux règles syntaxiques du langage XML.
    - Ils sont correctement traités par un parseur XML.
    - Document bien formé = document correct.
  - Les documents valides
    - Un document valide est un document XML bien formé et conforme à une DTD (Document Type Definition).
    - Le processeur XML vérifie le document par rapport à sa grammaire décrite dans la DTD.
    - Le document doit respecter les éléments, les attributs et les entités déclarés dans la DTD.

Champs d'application de XML (1)

- Messagerie
  - XML en tant que format standard pour l'échange de données
    - par exemple : XML/EDI
- Traitement
  - Communication client/serveur
    - par exemple : Transactions entre banques, commerce électronique, …
- Recherche
  - Recherche intelligente d'information
  - Web sémantique

Champs d'application de XML (2)

- Publication et collaboratif
  - Présentation variable de l'information
    - Avec HTML, le rendu est le même pour tout le monde
    - Avec XML, le choix de la restitution peut revenir au client.
  - L'auteur décide du contenu, le lecteur de la présentation.
    - Publication dans différents formats avec XSL (eXtensible StyleSheet Language)

- Intégration de système
  - Échanges automatisés
    - RSS (Really Simple Syndication)
  - EAI : Enterprise Application Integration
    - Architecture informatique permettant à des applications hétérogènes de gérer leurs échanges

Les structures de données de XML (1)

- Structure d'un document
  - Un document XML est toujours composé de trois éléments:
    - Le prologue,
    - un arbre d'éléments
    - des commentaires et instructions de traitements

Les structures de données de XML(2)


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE exemple SYSTEM "exemple.dtd">

<DEVIS>
   <!-- Entête -->
   <IDENTIFIANT>0401</IDENTIFIANT>
   <LIBELLE>Achats Réseaux</LIBELLE>
   <STATUT>Demande d'achat</STATUT>
   <FOURNISSEUR>Alcatel Commutation</FOURNISSEUR>
   <MONTANT devise="FRF">1452805.30</MONTANT>
   <DATE_LIVRAISON>12/12/2000<DATE_LIVRAISON>
   <!-- Lignes article -->
   <ARTICLE>
      <REFERENCE>ISML43S904</REFERENCE>
      <QUANTITE>280</QUANTITE>
      <PRIX_UNITAIRE devise="FRF">408.00</PRIX_UNITAIRE>
      <IMAGE href="media/ISML43S904.gif"/>
   </ARTICLE>
   ...
</DEVIS>
```

Entête

Elément racine

Elément

Commentaire

Attribut

Elément vide

Contenu

Les structures de données de XML (3)

- Usage des attributs
  - Aucune règle valable
- Exemple :

```
- <personne>
    <nom>Dupond</nom>
    <prenom>Jean</prenom>
  - <bureau site="Sophia">
      <bureau-nbr>2506</bureau-nbr>
      <telephone>+33.4.9365.7777</telephone>
      <fax>+33.4.9365.7788</fax>
    </bureau>
  </personne>
- <personne>
    <nom>Michard</nom>
    <prenom>Alain</prenom>
  - <bureau site="Rocquencourt">
      <bureau-nbr>3276</bureau-nbr>
      <telephone>+33.1.3963.7777</telephone>
      <fax>+33.1.3963.5114</fax>
    </bureau>
  </personne>
```

```
<personne name="Michard" prenom="Alain" site-name="Rocquencourt" telephone="01.3963.7777" bureau-nbr="2512" />
<personne name="Dupond" prenom="Jean" site-name="Sophia" telephone="04.9365.7777" bureau-nbr="2506" />
```

Langage XML :Les règles syntaxiques

Règle 1 : commencer par une déclaration XML

- **<?xml version="**1.0**" encoding="**iso-8859-1|UTF-8|…**" standalone="**yes|no**"?>**

- **Version :** XML est dans sa version 1.0 ou 1.1
- **Encoding** : décrire le codage utilisé dans le document. **Iso-8859-1** : latin occidental; **UTF-8** : Unicode; …
- **Standalone** : si le document est traité seul = **Yes**, s'il dépend d'un autre fichier (une DTD par exemple) = **No**

Langage XML :Les règles syntaxiques

Règle 2 : A l'exception des élément vides, un élément a toujours une balise d'ouverture et de fermeture.

- Un élément se compose de:
  - une balise d'ouverture
  - un contenu
  - une balise de clôture
- Exemple
  - <nom>contenu de l'élément</nom>
  - Nom =(caractères alphanumériques + '_' + '-' + '.')*
- Si l'élément est vide, il peut s'écrire uniquement avec la balise d'ouverture avec un slash avant le chevron fermant
  - Ex: <position/>

Les structures de données de XML (6)

Règle 3 : Un document XML a une seule racine qui inclut des éléments

- Un élément peut contenir d'autres éléments imbriqués
- Exemple :

```
<annuaire>
        <personne>
                <nom>Pillou</nom>
                <prenom>Jean-François</prenom>
        </personne>
        <personne>
                <nom>VanHaute</nom>
                <prenom>Nico</prenom>
        </personne>
</annuaire>
```

Langage XML :Les règles syntaxiques

Règle 4 : respecter la case

- **<Auteur>** ≠ **<auteur>** ≠ **<AUTEUR>**

- Les noms des éléments doivent respecter les majuscules et les minuscules

- Les balises de début et de fin correspondantes doivent utiliser la même capitalisation : <auteur> </auteur>

Langage XML : Les règles syntaxiques

Règle 5 : les valeurs des attributs doivent être "quottés"

- Toutes les valeurs des attributs sont spécifiées entre une paire de guillemets doubles.
- Exemple :

  <Docteur NE="1947" MORT="2013">Abderrahman Smit </Docteur>

  - Ne pas inclure les caractères: ^ % &
  - Entités prédéfinies
    - < pour le caractère <
    - > pour le caractère >
    - & pour le caractère &
    - ' pour le caractère `
    - " pour le caractère «

Langage XML : Les règles syntaxiques

Règle 6 : Toute balise ouverte doit être fermée sans chevauchement des éléments

- En XML, toute balise ouverte doit être obligatoirement fermée.
- Un élément qui contient des sous-éléments doit les contenir en totalité, balise ouvrante et fermante.
- Un élément principal (la racine) doit encapsuler tous les autres éléments.
- Il faut toujours respecter la structure hiérarchique du document.

Langage XML : Les règles syntaxiques

Règle 7 : les noms des éléments (balises)

- Les noms peuvent contenir des lettres, des chiffres ou d'autres caractères.
- Les noms ne peuvent débuter par un nombre ou un signe de ponctuation.
- Les noms ne peuvent pas commencer par les lettres xml.
- Les noms ne peuvent contenir des espaces.
- La longueur des noms est libre mais on conseille de rester raisonnable.
- On évite certains signes qui pourraient selon les logiciels, prêter à confusion comme "-", ";", ".", "<", ">", etc

Langage XML : Les règles syntaxiques

Règle générale

- Le XML est un langage strict.
- Un document XML doit impérativement respecter la syntaxe du XML. Le document XML est "bien formé" [Well-formed].
- Seuls les documents "bien formés" seront affichés correctement. A la moindre erreur de syntaxe, le document ne sera pas ou ne sera que partiellement affiché.

Plan

- Langage XML
- DTD
  - Définitions
  - Déclaration d'éléments simples
  - Déclaration d'éléments composés
  - Déclaration d'attributs
  - Exemples

DTD (Document Type Definition) (1)

- DTD : Une structure de données qui définit des règles à suivre dans la création d'un document XML

- La DTD permet de définir son propre langage basé sur XML
  - Vocabulaire
  - Grammaire

DTD (2)

- **Document bien formé** (Well Formed document)
  - balises correctement imbriquées
  - Analysable (parsable) et manipulable
  - pas nécessairement valide par rapport à la DTD
- **Document valide** (Valid document)
  - bien formé + conforme à la DTD

DTD (3)

- DTD
  - Permet de définir le «vocabulaire» et la structure qui seront utilisés dans le document XML.
  - Grammaire du langage dont les phrases sont des documents XML (instances).
  - Peut être mise dans un fichier (DTD externe) et être appelée dans le document XML

Déclaration d'éléments simples (1)

- <! ELEMENT *balise* (*définition*) >
  - Le paramètre *définition* représente soit un type de données prédéfini, soit un type de données composé, constitué lui même d'éléments.
  - Types prédéfinis
    - **ANY** : L'élément peut contenir tout type de donnée
    - **EMPTY** : L'élément ne contient pas de données spécifiques
    - **(#PCDATA)** : L'élément doit contenir une chaîne de caractère
  - Exemple
    - <! ELEMENT Nom (#PCDATA)>
    - <Nom>Victor Hugo</Nom>

Déclaration d'éléments simples (2)

- Exemple
  - <!ELEMENT tutorial (#PCDATA)>
  - <tutorial>Ceci est un document XML</tutorial>
    - Valide!
  - <tutorial/>
    - Valide
  - <text>Ceci est un document XML</text>
    - Non valide

Déclaration d'éléments composés (1)

- Élément composé d'une séquence ou d'un choix d'éléments.
- Syntaxe spécifique avec opérateurs de composition d'éléments :
  - <! ELEMENT balise (*composition*) >

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A **ou** B peuvent être présents (pas les deux)	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Déclaration d'élément composé (2)

- **Exemple 1:**

  - <!ELEMENT XX (AA , BB)>
    <!ELEMENT AA (#PCDATA)>
    <!ELEMENT BB (#PCDATA)>
  - <XX> <AA>Début</AA> <BB>Fin</BB> </XX>
    - Valide
  - <XX> <AA/> <BB/> </XX>
    - Valide
  - <XX> <AA/> </XX>
    - Non valide
  - <XX> <BB/> <AA/> </XX>
    - Non valide

Déclaration d'éléments composés (3)

- **Exemple 2 :**
  - <!ELEMENT XX (AA* , BB)>
    <!ELEMENT AA (#PCDATA)>
    <!ELEMENT BB (#PCDATA)>
  - <XX> <AA/> <BB/> </XX>
    - Valide
  - <XX> <BB/> </XX>
    - Valide
  - <XX> <AA/> <AA/> <AA/> <AA/> <AA/> <AA/> <AA/> <BB/> </XX>
    - Valide
  - <XX> </XX>
    - Non valide
  - <XX> <BB/> <AA/> </XX>
    - Non valide : L'élément BB doit suivre l'élément AA
  - <XX> <AA/> <AA/> <AA/> <AA/> <BB/> <AA/> <AA/> </XX>
    - Non valide

Déclaration d'éléments composés (4)

- **Exemple 3 :**
  - <!ELEMENT XX (AA? , BB+)>
    <!ELEMENT AA (CC? , DD*)>
    <!ELEMENT BB (CC , DD)>
    <!ELEMENT CC (#PCDATA)>
    <!ELEMENT DD (#PCDATA)>
  - <XX> <AA> <CC/><DD/> </AA> <BB> <CC/><DD/> </BB> </XX>
    - Valide
  - <XX> <BB> <CC/><DD/> </BB> </XX>
    - Valide
  - <XX> <AA> <CC/><CC/> <DD/><DD/> </AA> <BB> <CC/><DD/> </BB> </XX>

    Non valide : L'élément AA peut contenir un élément CC au maximum

Déclaration d'éléments composés (5)

- **Exemple 4 :**

```
<!ELEMENT personne (nom, prenom+, tel?, adresse)>
        <!ELEMENT nom (#PCDATA) >
        <!ELEMENT prenom (#PCDATA) >
        <!ELEMENT tel (#PCDATA) >
        <!ELEMENT Adresse  ANY >
<personne>
        <nom>Hugo</nom>
        <prenom>Victor</prenom>
        <prenom>Charles</prenom>
        <tel>01</tel>
        <adresse>
                <rue/>
                <ville>Paris</ville>
        </adresse>
</personne>
```

Déclaration d'attributs (5)

- <! ATTLIST balise Attribut Type Mode >
  - Balise : spécifie l'élément auquel est attaché l'attribut
  - Attribut : est le nom de l'attribut déclaré
  - Type : définit le type de donnée de l'attribut choisi parmi:
    - **CDATA**
      - Chaînes de caractères entre guillemets ("aa") **non analysées**
    - **Enumération**
      - Liste de valeurs séparées par |
        - <! ATTLIST balise Attribut (Valeur1 | Valeur2 | ...) >
    - **ID et IDREF**
      - Clé et référence à une clé
  - Mode : précise le caractère obligatoire ou non de l'attribut
    - #REQUIRED, #IMPLIED ou #FIXED

Exemple d'attributs

- **Exemple 1 :**

```
<! ATTLIST personne
                num ID,
                age CDATA,
                genre (Masculin | Feminin ) >
```

- **Exemple 2 :**

```
<!ELEMENT etudiant (#PCDATA) >
<!ATTLIST etudiant
                cycle (Licence| Ingénieur ) #REQUIRED
                spécialité CDATA #IMPLIED>
```

- **Exemple 3 :**

```
<!ELEMENT editeur (#PCDATA) >
<!ATTLIST editeur  ville CDATA #FIXED "Paris">
```

Exemple de DTD

- **Exemple 4 :**

```
<!ELEMENT doc (livre* | article+) >
<!ELEMENT livre (titre, auteur+) >
<!ELEMENT article (titre, auteur*) >
<!ELEMENT titre(#PCDATA) >
<!ELEMENT auteur(nom, adresse) >
        <!ATTLIST auteur id ID #REQUIRED >

<!ELEMENT nom(prenom?, nomfamille) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT nomfamille (#PCDATA) >
<!ELEMENT adresse ANY >
```

Exemple de DTD interne


```
<?XML version="1.0" standalone= " yes"?>
<!DOCTYPE classe [
<!ELEMENT classe (etudiant+)>

<!ELEMENT etudiant (competences+, specialite, age)>
<!ATTLIST etudiant NOM CDATA #REQUIRED>

<!ELEMENT competences EMPTY>
<!ATTLIST competences info CDATA #REQUIRED>

<!ELEMENT specialite (#PCDATA)>
<!ELEMENT age (#PCDATA)>
]>
<classe>
  <etudiant NOM="Yahya">
    <competences info="AOS"/>
    <specialite >Réseau</specialite >
    <age>22</age>
  </etudiant >
  ……..
</ classe >
```

Exemple de ID et IDREF


```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
        <!ELEMENT DOCUMENT(PERSONNE*)>
        <!ELEMENT PERSONNE (#PCDATA)>
                <!ATTLIST PERSONNE PNUM ID #REQUIRED>
                <!ATTLIST PERSONNE MERE IDREF #IMPLIED>
                <!ATTLIST PERSONNE PERE  IDREF #IMPLIED>
]>

 <DOCUMENT>
      <PERSONNE PNUM = "P1">Marie</PERSONNE>
      <PERSONNE PNUM = "P2">Jean</PERSONNE>
      <PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Pierre</PERSONNE>
      <PERSONNE PNUM = "P4" MERE="P1" PERE="P2">Julie</PERSONNE>
 </DOCUMENT>
```

DTD externe

- Modèle pour plusieurs documents
  - partage des balises et structures
- Définition locale ou externe
  - <!DOCTYPE doc SYSTEM "doc.dtd">
  - <!DOCTYPE doc PUBLIC "www.myweb.com/doc.dtd">
- Exemple de document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE etudiant SYSTEM  "etudiant.dtd">
...
```

Entité paramètre

- Permet la définition d'un groupe d'éléments sous un nom (macro)
  - <!ENTITY % nom "definition">
- Réutilisable dans une DTD par un simple appel :
  - %nom;
  - Exemple :
    - <!ENTITY % cycles "(licence | ingénieur)">
      <!ATTLIST auteur cycle %**cycles;** #REQUIRED>
- Peut être externe :
  - <!ENTITY %mpeg SYSTEM "http://www.myweb.fr">

Entité générale

- Permet la définition d'un texte sous un nom <!ENTITY nom "texte">

- Réutilisable dans un document par simple appel :

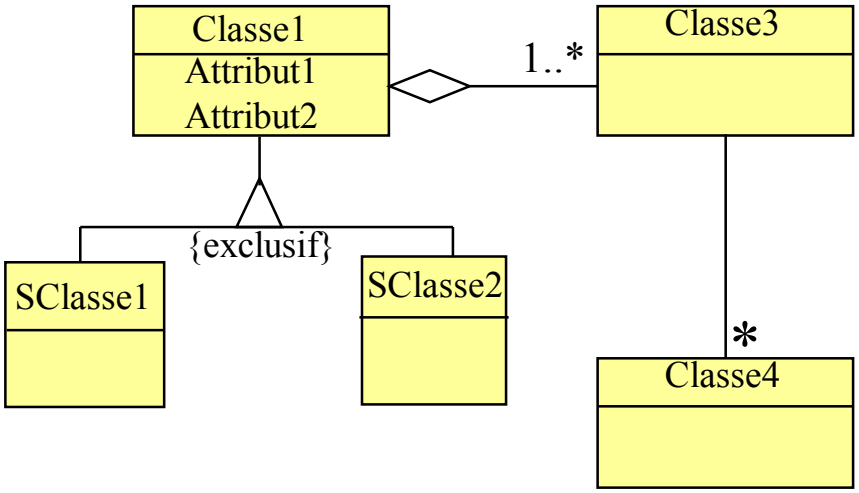
```
<?xml version="1.0"?>
<!DOCTYPE exemple [
    <!ELEMENT exemple (#PCDATA, important)>
    <!ELEMENT important (#PCDATA)>
    <!ENTITY cie "Réunion">
    <!ENTITY imp "<important>Attention!</important>"> ]>

<exemple> &cie; &imp;
 </exemple>
```

DTD: Quelques règles d'écriture

- Modularité
  - définir dans des entités séparées de parties réutilisables.
- Précédence
  - Regrouper les déclarations d'entités en entête.
- Abstraction
  - Utiliser des entités pour les modèles de contenu.
- Spécificité
  - Éviter les DTD trop générales.
- Simplicité
  - Découper les DTD trop complexes.

DTD > Comment concevoir une DTD ?

- Méthode UML-XML
  - décrire les sources de données avec UML
  - utilisation de :
    - classe
    - attribut
    - agrégation
    - association
    - généralisation
  - fixer les cardinalités
    - associations 0..* ou 1..* (++)
    - attributs 0 ou 1 (?)



Exercice

- Définir les DTD pour publier une BD

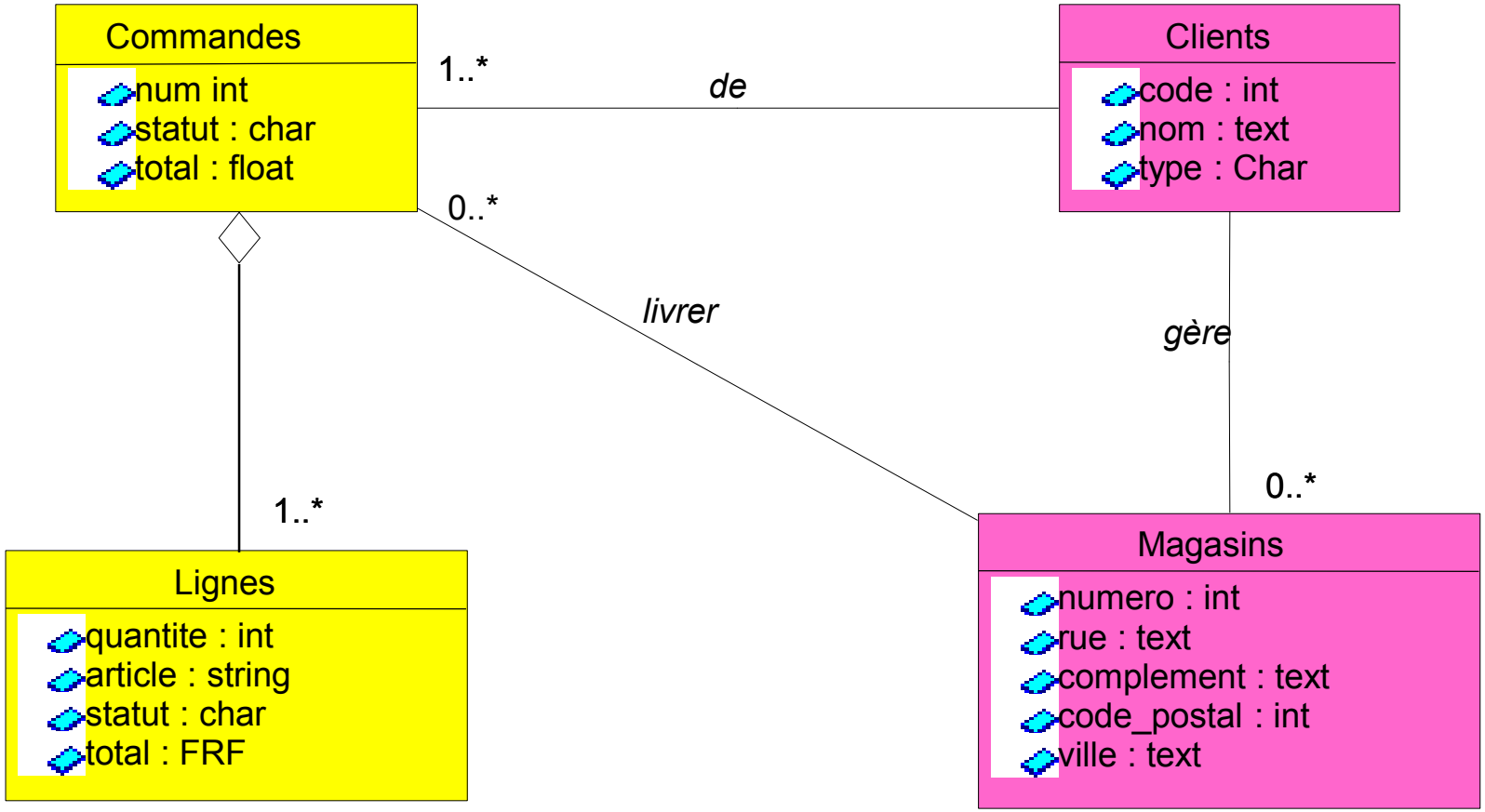


Schéma UML

DTD > Définition des types et classes


```
<!-- Types de base-->
<!ENTITY % int "(#PCDATA)">
<!ENTITY % float "(#PCDATA)">
<!ENTITY % char "(#PCDATA)">
<!ENTITY % string "(#PCDATA)">

<!-- Classe Commande -->
<!ELEMENT Commande (ligne+ )>
<!ATTLIST Commande NUM ID #REQUIRED>
<!ATTLIST Commande cstatut %char;  #REQUIRED>
<!ATTLIST Commande ctotal % float;  #REQUIRED>
<!-- Classe Ligne -->
<!ELEMENT Ligne (article, quantite, statut?, total?)>
<!ELEMENT article %string;>
<!ELEMENT quantite %int;>
<!ELEMENT lstatut %char;>
<!ELEMENT ltotal %float;>
```