Le tableau 13-24 décrit la génération d'une arborescence XML décrivant les affrètements ordonnés par compagnie.

Tableau 13-24 Génération de contenus XML

Code SQL

```
SELECT XMLElement
 ("Affretement",
  XMLAttributes(c.codec AS "comp"),
  XMLForest(c.nomCompa AS "nomComp"),
  XMLElement("Vols",
    (SELECT XMLAgg(XMLElement
    ("Vol",
     XMLForest(
         TO_CHAR(af.date_a,'DD/MM/YYYY')
                    AS "date",
         av.typav   AS "avion",
         af.nb_passagers AS "passagers")))
    FROM  affreter_R af, avion_R av
    WHERE af.codec = c.codec
    AND  av.na = af.na) )) AS "Contenu_XML"
 FROM compagnie_R c ORDER BY nomCompa;
```

Résultat

```
Contenu_XML
--------------------------------
<Affretement comp="AB">
  <nomComp>Air Blagnac</nomComp>
  <Vols></Vols>
</Affretement>
<Affretement comp="AF">
  <nomComp>Air France</nomComp>
  <Vols>
   <Vol>
     <date>08/12/2007</date>
     <avion>A320</avion>
     <passagers>150</passagers>
   </Vol>
   <Vol>
     <date>08/12/2007</date>
     <avion>A318</avion>
     <passagers>130</passagers>
   </Vol>
   <Vol>
      <date>09/12/2007</date>
      <avion>A318</avion>
      <passagers>110</passagers>
   </Vol>
  </Vols>
</Affretement>
<Affretement comp="EJ">
  <nomComp>Easy Jet</nomComp>
  <Vols>
   <Vol>
    <date>08/12/2007</date>
    <avion>A320</avion>
    <passagers>120</passagers>
   </Vol>
  </Vols>
</Affretement>
```

Vues XMLType

Concernant les données qui sont stockées dans des tables relationnelles ou objet-relationnelles, les vues `XMLType` permettent de composer du contenu XML contraint ou pas par une grammaire préalablement enregistrée.

Sans grammaire

Le tableau 13-25 présente la déclaration et l'interrogation de la vue `XMLType` qui fusionne des données des trois tables relationnelles précédentes. La requête de définition inclut en plus un identifiant objet (par exemple ici, le nom de la compagnie). Les extractions retournent le nombre d'affrètements stockés puis le détail d'un affrètement.

Tableau 13-25 Vue XMLType

Création de la vue

```
CREATE VIEW compagnie_xml OF XMLType
  WITH OBJECT ID
 (SUBSTR(EXTRACTVALUE(
   OBJECT_VALUE,'/Affretement/nomComp'),
   1,30))
 AS SELECT XMLElement("Affretement",
    XMLAttributes(c.codec AS "comp"),
    XMLForest(c.nomCompa AS "nomComp"),
    XMLElement("Vols",
    (SELECT XMLAgg(XMLElement("Vol",
     XMLForest(TO_CHAR(af.date_a,
        'DD/MM/YYYY') AS "date",
        av.typav AS "avion",
        af.nb_passagers AS "passagers")))
     FROM  affreter_R af, avion_R av
     WHERE af.codec = c.codec
     AND   av.na = af.na)) AS "Contenu_XML"
 FROM compagnie_R c;
```

Interrogations de la vue

```
SELECT COUNT(OBJECT_VALUE)
       FROM compagnie_xml ;
COUNT(OBJECT_VALUE)
-------------------
                  3

SELECT OBJECT_VALUE
  FROM compagnie_xml
  WHERE EXISTSNODE(OBJECT_VALUE,
  '/Affretement[@comp="EJ"]')=1;
OBJECT_VALUE
--------------------------------
<Affretement comp="EJ">
<nomComp>Easy Jet</nomComp>
 <Vols>
  <Vol>
    <date>08/12/2007</date>
    <avion>A320</avion>
    <passagers>120</passagers>
  </Vol>
 </Vols>
</Affretement>
```

À partir d'un type objet

La fonction `SYS_XMLGEN` génère une instance `XMLType` à partir d'un type objet. Le tableau 13-26 décrit la création d'un type objet décrivant les affrètements (en utilisant un attribut). La requête de définition de la vue `XMLType` (tous les affrètements de plus de 110 passagers dans un format XML) fait intervenir la fonction `XMLFORMAT` qui compose l'élément racine.

Tableau 13-26 Vue XMLType à partir d'un type

Création du type et de la vue	Interrogation de la vue
`CREATE TYPE affreter_type AS OBJECT ("@na" CHAR(6), codec VARCHAR(6), date_a DATE, nb_passagers NUMBER); /`	`SELECT OBJECT_VALUE FROM affreter_view_xml WHERE EXISTSNODE(OBJECT_VALUE, 'Affretement/CODEC[text()="EJ"]')=1;`
`CREATE VIEW affreter_view_xml OF XMLType WITH OBJECT ID (EXTRACT(OBJECT_VALUE, '/Affetement/@na').GETNUMBERVAL()) AS SELECT SYS_XMLGEN(affreter_type(a.na, a.codec, a.date_a, a.nb_passagers), XMLFORMAT('Affretement')) FROM affreter_R a WHERE a.nb_passagers > 110;`	`OBJECT_VALUE ----------<?xml version="1.0"?> <Affretement na="F-GODF"> <CODEC>EJ</CODEC> <DATE_A>08/12/07</DATE_A> <NB_PASSAGERS>120</NB_PASSAGERS> </Affretement>`

Association d'une grammaire

Une vue `XMLType` peut être associée à une grammaire pour contraindre davantage les données extraites. Considérons la simple grammaire caractérisant l'élément `avioncomp` et définissons une vue `XMLType` peuplée à partir des données (tous les avions) des tables relationnelles.

Tableau 13-27 Structure et grammaire de la vue

Structure XML	Grammaire
`<?xml version="1.0" encoding="ISO-8859-1"?> <avioncomp na = "…"> <nomAv> … </nomAv> <capacite> … </capacite> <compav> <comp> … </comp> <nomcomp> … </nomcomp> </compav> </avioncomp>`	`<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <xsd:element name="avioncomp" type="avionType"/> <xsd:complexType name="avionType"> <xsd:sequence> <xsd:element name="nomAv" type="xsd:string"/> <xsd:element name="capacite" type="xsd:int"/> <xsd:element name="compav" type="compavType"/> </xsd:sequence> <xsd:attribute name="na" type="xsd:string"/> </xsd:complexType> <xsd:complexType name="compavType"> <xsd:sequence> <xsd:element name="comp" type="xsd:string"/> <xsd:element name="nomcomp" type="xsd:string"/> </xsd:sequence> </xsd:complexType> </xsd:schema>`

Le tableau 13-28 décrit la définition et l'extraction complète de la vue. La fonction GETNUMBERVAL permet d'affecter une valeur numérique à chaque enregistrement extrait et définir ainsi avec WITH OBJECT ID l'identifiant de la vue. On retrouve les fonctions de la norme ANSI qui génèrent du contenu XML.

Tableau 13-28 Vue XMLType associée à une grammaire

Création de la vue

```
CREATE VIEW avicomp_view_xml OF XMLType
  XMLSCHEMA
  "http://www.soutou.net/Avioncomps.xsd"
  ELEMENT "avioncomp" WITH OBJECT ID
(EXTRACT (OBJECT_VALUE,
      '/AvionComp/@immat').GETNUMBERVAL())
  AS SELECT XMLElement
  ("AvionComp",
   XMLAttributes(av.na AS "immat"),
   XMLForest(av.typav AS "nomav",
             av.capacite AS "nbplaces"),
   XMLElement
  ("compav", XMLForest(c.codec AS "comp",
            c.nomcompa AS "nomcomp")))
  FROM avion_R av, compagnie_R c
  WHERE av.proprio = c.codec;
```

Contenu de la vue

```
SELECT OBJECT_VALUE
      FROM avicomp_view_xml;
OBJECT_VALUE
--------------------------------
<AvionComp immat="F-GODF">
  <nomav>A320</nomav>
  <nbplaces>170</nbplaces>
  <compav>
    <comp>AB</comp>
    <nomcomp>Air Blagnac</nomcomp>
  </compav>
</AvionComp>
<AvionComp immat="F-PROG">
  <nomav>A318</nomav>
  <nbplaces>140</nbplaces>
  <compav>
    <comp>AF</comp>
    <nomcomp>Air France</nomcomp>
  </compav>
</AvionComp>
<AvionComp immat="F-WOWW">
  <nomav>A380</nomav>
  <nbplaces>490</nbplaces>
  <compav>
    <comp>EJ</comp>
    <nomcomp>Easy Jet</nomcomp>
  </compav>
</AvionComp>
```

Génération de grammaires annotées

La fonction GENERATESCHEMA du paquetage DBMS_XMLSCHEMA permet de générer une grammaire annotée *XML Schema*. Les paramètres sont à inscrire en majuscules. Ils décrivent le nom du schéma d'Oracle qui contient le type objet-relationnel et le nom du type lui-même. Générons la grammaire annotée décrivant le type societe_type.

Il restera à ajouter d'éventuelles annotations qui contraindront ou préciseront le stockage des sociétés. Ce mécanisme fonctionne également pour les collections objet (AS TABLE OF).

Tableau 13-29 Génération d'une grammaire annotée

Web

Code SQL	Résultats (en partie)

CREATE TYPE adresse_type
  AS OBJECT (nrue CHAR(3),
  nomrue VARCHAR2(20),
  ville VARCHAR2(20))
/
	<?xml version="1.0"?>
	<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:xdb="http://xmlns.oracle.com/xdb"
	xsi:schemaLocation="http://xmlns.oracle.com/xdb
	http://xmlns.oracle.com/xdb/XDBSchema.xsd">

CREATE TYPE societe_type
  AS OBJECT (siret VARCHAR2(15),
  creation DATE,
  adresse_t adresse_type,
  effectif NUMBER)
/
	<xsd:element name="SOCIETE_TYPE" type="SOCIETE_TYPEType"
	xdb:SQLType="SOCIETE_TYPE" xdb:SQLSchema="SOUTOU"/>
	
	<xsd:complexType name="SOCIETE_TYPEType"
	xdb:SQLType="SOCIETE_TYPE" xdb:SQLSchema="SOUTOU"
	xdb:maintainDOM="false">
	<xsd:sequence>
	<xsd:element name="SIRET" xdb:SQLName="SIRET"
	xdb:SQLType="VARCHAR2">
	<xsd:simpleType>
	<xsd:restriction base="xsd:string">
	<xsd:maxLength value="15"/>
	</xsd:restriction>
	</xsd:simpleType>
	</xsd:element>
	<xsd:element name="CREATION" type="xsd:date"
	xdb:SQLName="CREATION" xdb:SQLType="DATE"/>
	<xsd:element name="ADRESSE_T" type="ADRESSE_TYPEType"
	xdb:SQLName="ADRESSE_T"
	xdb:SQLSchema="SOUTOU" xdb:SQLType="ADRESSE_TYPE"/>
	<xsd:element name="EFFECTIF" type="xsd:double"
	xdb:SQLName="EFFECTIF" xdb:SQLType="NUMBER"/>
	</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="ADRESSE_TYPEType"
	xdb:SQLType="ADRESSE_TYPE" xdb:SQLSchema="SOUTOU"
	xdb:maintainDOM="false">
	<xsd:sequence>
	<xsd:element name="NRUE" xdb:SQLName="NRUE"
	xdb:SQLType="CHAR">
	<xsd:simpleType>
	<xsd:restriction base="xsd:string">
	<xsd:length value="3"/>
	</xsd:restriction>
	</xsd:simpleType>
	</xsd:element>
	…
	</xsd:schema>

SELECT
  DBMS_XMLSCHEMA.**GENERATESCHEMA**(
  'SOUTOU','SOCIETE_TYPE')
  FROM DUAL;	

Dictionnaire des données

Le dictionnaire des données propose un certain nombre de vues (préfixées par `USER` pour les objets du schéma courant, `ALL` pour les objets sur lesquels on a des privilèges et `DBA` pour tous les objets et quel que soit le schéma) qui intéresseront les utilisateurs de XML DB.

Tables XMLType

Au niveau d'un utilisateur, la vue `USER_XML_TABLES` décrit les tables `XMLType` en ce qui concerne le type de stockage et les options de grammaire.

Tableau 13-30 Nature des tables XMLType

Code SQL

```
SELECT TABLE_NAME,
  XMLSCHEMA,
  STORAGE_TYPE
  FROM USER_XML_
  TABLES;
```

Résultats

```
TABLE_NAME
 ----------------------------
XMLSCHEMA                                STORAGE_TYPE
---------------------------------------- --------------
AVION_OR_XMLSCHEMA
http://www.soutou.net/avions.xsd         OBJECT-RELATIONAL

COMPAGNIE_BINARYXML_GRAMMAIRE
http://www.soutou.net/compagnies3.xsd BINARY

COMPAGNIE_OR_XMLSCHEMA
http://www.soutou.net/compagnies.xsd  OBJECT-RELATIONAL
```

Code SQL

```
SELECT TABLE_NAME,
  ELEMENT_NAME,
  ANYSCHEMA,
  NONSCHEMA
  FROM USER_XML_
  TABLES;
```

Résultats

```
TABLE_NAME                     ELEMENT_NAME     ANY NON
------------------------------ ---------------- --- ---
COMPAGNIE_BINARYXML                             NO  YES
AVION_OR_XMLSCHEMA             avion
COMPAGNIE_BINARYXML_GRAMMAIRE  compagnie        NO  NO
COMPAGNIE_OR_XMLSCHEMA         compagnie
```

Colonnes XMLType

Sur le même principe, la vue `USER_XML_TAB_COLS` décrit les colonnes `XMLType` en ce qui concerne le type de stockage et la grammaire.