Chapitre IX

L'intégration de données

Le problème

De façon très générale, le problème de *l'intégration de données* (*data integration*) est de permettre un accès cohérent à des données d'origine, de structuration et de représentation différentes.

Nous allons étudier deux techniques liées à ce problème.

- Les *entrepôts de données* (*data warehouses*) qui visent à regrouper des données en vue de leur analyse statistique et de leur exploitation en gestion stratégique.
- Le langage XML (*eXtended Markup Language*) qui définit un cadre générique de représentation de données (semi-)structurées.

Les entrepôts de données (Data Warehouses)

Motivation

- L'information accumulée dans les *bases de données opérationnelles* (exploitées pour la gestion quotidienne) est aussi utile pour une gestion plus "stratégique" de l'entreprise.
- Le type de requêtes nécessaires (e.g. "classer les clients en fonction de la fréquence de leur demande d'interventions") nécessite un traitement
  - lourd, et donc ayant un impact sur les performances de la base de données opérationnelle,
  - pas toujours aisément expressible en SQL,
  - et impliquant parfois des données se trouvant dans des BD distinctes.
- Il est donc intéressant pour effectuer ce type de traitement de consolider les données dans une base de données spécifiquement conçue à cet effet : un entrepôt de données.

Qu'est-ce qu'un entrepôt de données

Un entrepôt de données est une base de données

- Consolidant les données de bases de données opérationnelles,
- Utilisée en consultation et seulement mise à jour périodiquement,
- Organisée pour permettre le traitement de requêtes "analytiques" plutôt que transactionnelles" (OLAP par rapport à OLTP).
  - OLTP : On-line transaction processing. Petites transactions consistant en une recherche d'informations et, souvent, une mise à jour.
  - OLAP : On-line analytical processing. Grosses transactions impliquant une fraction importante des données réalisant, par exemple, un calcul statistique.

De la base de données opérationnelle à l'entrepôt de données

Les informations contenues dans l'entrepôt de données sont toutes issues des bases de données opérationnelles, mais

- leur schéma est en général différent,
- les mises à jour ne se font que périodiquement (par exemple tous les jours).

On appelle *extraction* le processus par lequel les données opérationnelles sont transférées vers l'entrepôt.

- Le schéma de l'entrepôt peut être considéré comme une *vue* sur les données opérationnelles ; on parle de vue *matérialisée* vu que les relations correspondant à la vue sont effectivement créées.
- Il est utile, mais parfois difficile, de pouvoir mettre à jour les données de l'entrepôt sans régénérer l'entièreté de celui-ci ; on parle de mise à jour incrémentale.

Exemple

Utilisation d'un entrepôt par une entreprise de distribution.

- Les données de vente sont enregistrées dans les différents magasins (OLTP).
- Chaque nuit, les données des différents magasins sont transférées dans un entrepôt de données au siège de la firme.
- Les données de l'entrepôt sont utilisée pour mettre au point des stratégies commerciales, des campagnes de promotion . . .

L'organisation des données d'une entrepôt : les schémas en étoile

On distingue habituellement deux types de données dans un entrepôt.

1. Les *faits* : grosse accumulation de données reprenant des faits simples, par exemple des chiffres de ventes ;
2. Les *données "dimensionnelles"* : données en quantité réduite, souvent statiques qui précisent des informations sur les éléments apparaissant dans les faits.

Exemple de schéma en étoile

On considère en entrepôt de données dans lequel on accumule des informations concernant la consommation de bières dans les bars d'une chaîne.

– Les faits sont accumulés dans une relation dont le schéma pourrait être

  `vente(bar, biere, consommateur, date, heure, prix)`

– Les données dimensionnelles pourraient être représentées dans des relations dont les schémas seraient les suivants :

  `bars(bar, adresse, gerant)`
  `bieres(biere, taux_alcool, fabriquant)`
  `consommateurs(consommateur, adresse, date_naissance)`

Attributs de “dimension” et attributs dépendants

On distingue deux types d'attributs dans la table des faits :

– les *attributs de dimension*, les clés des relations “dimensionnelles” et

– les *attributs dépendants*, les valeurs déterminées par les attributs de dimension.

Exemple : dans la relation `vente`, l'attribut `prix` est un attribut dépendant déterminé par les attributs `bar, biere, consommateur, date, heure`.

Comment extraire des informations d'un entrepôt ?

Il y a deux approches principales :

1. Exprimer les requêtes en SQL, on parle alors de ROLAP (relational on-line analytical processing).
2. Voir l'entrepôt comme une base de données *multidimensionnelle*, dont le modèle est un cube à n-dimensions. Il y a une dimension du cube par attribut dimensionnel et les éléments du cube sont les valeurs des attributs dépendants. On parle alors de MOLAP (multidimensional on-line analytical processing).

Les techniques ROLAP

Une requête ROLAP est en général exprimée comme suit.

1. On calcule le joint de la table des faits et des relations dimensionnelles ;
2. On sélectionne des tuples en fonctions des données dimensionnelles ;
3. On groupe ces données suivant certaines dimensions ;
4. On calcule une valeur agrégée (le plus souvent une somme).

Un exemple de requête ROLAP

Pour chaque bar d'Angleur, trouver la somme des ventes de chaque bière produite par Interbrew.

2. On filtre le joint de la table des faits et des tables dimensionnelles par `fabriquant` = Interbrew et `adresse` = Angleur ;
3. On groupe le résultat par `bar` et `biere` ;
4. On agrège en calculant la somme de `prix`.

ROLAP et problèmes d'implémentation

Les requêtes ROLAP peuvent être très coûteuses en ressources. Il y a des techniques qui permettent de les accélérer.

– *Les index "bitmaps"*. Il s'agit d'accélérer la sélection suivant les dimensions en créant pour chaque valeur des clés de dimension un vecteur de bits qui indique quels tuples de la table des faits ont cette valeur. Cela est très efficace pour gérer des sélections dimensionnelles multiples.

– *Les vues matérialisées*. On peut précalculer certains joints et ainsi créer des vues matérialisées.

L'approche MOLAP et les cubes de données (data cubes)

- Les clés des tables de dimension sont considérées comme les axes d'un hypercube.
- Les attributs dépendants apparaissent comme les points du cube.
- Le cube peut aussi contenir des valeurs agrégées (en général la somme) suivant les axes, plans, hyperplans, . . .du cube.

L'approche MOLAP : Exploiter un cube de données

Dans l'approche MOLAP, les requêtes consistent à effectuer les opérations suivantes.

- Sélectionner et agréger suivant certains axes (*roll-up*). Par exemple, grouper les bars par région. On parle aussi de *slicing* (sélectionner suivant une valeur) et de *dicing* (sélectionner suivant un domaine de valeurs).

- Désagréger certains groupements (*drill-down*). Par exemple, après avoir groupé les bières par taux d'alcool, les séparer par fabriquant.

- Eliminer certaines dimensions (projeter), ce qui se fait en remplaçant les points du cube par les agrégats correspondants. On parle de *pivoting*.

La fouille des données (Data Mining)

La fouille des données

- **Le concept.** Il s'agit de trouver des informations dans un entrepôt, allant au delà de ce que l'on peut aisément et efficacement exprimer avec une requête.

- **Exemple**. Dans une base de données de ventes,

  `ventes{id_panier,produit}`

  on essaie de trouver quels éléments apparaissent souvent simultanément. Le but est, par exemple, de cibler une offre promotionnelle ou de proposer des achats groupés.

La recherche des paires : Une approche naïve

- On se fixe un *seuil de support* (pourcentage de paniers dans lesquels une paire de produits apparaît) pour sélectionner les paires intéressantes.
- On calcule le joint de la relation `ventes` avec elle-même.
- On groupe par paires de produits.
- On sélectionne sur la fréquence.

Problème : coût prohibitif.
Solution : sélection *a priori* des articles isolés vendus fréquemment. Seuls ceux là peuvent apparaître dans les paires fréquentes.

Le langage XML

Origine

- L'origine se situe dans les langages d'annotation de textes, permettant d'ajouter à un texte des informations sur la nature de parties du texte ou des indications de mise en page.
- HTML est un tel langage d'annotation qui prévoit un ensemble prédéfini d'annotations possibles.
- SGML (Standard Generalized Markup Language) et XML qui en est une révision simplifiée permet d'utiliser un ensemble *définissable* d'annotations. SGML est assez ancien (standardisé en 1986), XML est plus récent (1998) et a en particulier été conçu pour être utilisé dans le cadre du Web.
- On peut voir HTML comme XML utilisé avec un ensemble particulier d'annotations.

XML et bases de données

- Dans une base de données, le schéma est fixé et conservé séparément des données. En XML, le schéma (précisé par les annotations) est incorporé dans les données.

- L'intégration du schéma dans les données permet beaucoup de flexibilité. Le schéma et le contenu d'une base de donnée peuvent donc facilement être représentés en XML, et ce indépendemment du type de base de données utilisé.

- XML est donc un format très utile pour exporter de l'information d'une base de données et l'incorporer dans une autre.

- C'est aussi une format utile pour extraire des informations d'une base de données et les fournir à une application, par exemple un traitement de texte.

XML : exemples d'utilisation

Le programme des cours de l'ULg

- L'information sur les cours et programmes de cours de l'ULg est maintenue dans une base de données.
- De cette base de données, on extrait l'information dans le format XML.
- L'information sous format XML est utilisée pour produire d'une part le programme accessible par le Web (format HTML) et la version imprimée du programme de cours.

Les documents sauvegardés par "Open office"

- Un fichier "open office" utilisé pour sauvegarder par exemple un document est une collection de fichiers XML comprimés.

XML : exemples d'utilisation - Open Office

```
pwportb% ls -l
total 8
-rw-r--r--    1 pw       pw-adm       5495 Apr 15 15:37 test.sxw
pwportb% unzip test.sxw
Archive:  test.sxw
 extracting: mimetype
  inflating: content.xml
  inflating: styles.xml
 extracting: meta.xml
  inflating: settings.xml
  inflating: META-INF/manifest.xml
pwportb% cat content.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE office:document-content
PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN" "office.dtd">

......
```

La structure d'un document XML : un exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<a>Latour</a>
<de>Wolper</de>
<sujet>Rappel</sujet>
<texte>Ne pas oublier l'examen de BD!</texte>
</note>
```

– La première ligne déclare qu'il s'agit d'un document XML et donne le codage des caractères utilisé.

– Le reste sont des *éléments* compris entre une *annotation* (*tag*) d'ouverture (e.g.`<note>`) et de fermeture (e.g. `</note>`) et structurés sous forme d'arbre (un élément peut en inclure d'autres).

Les attributs d'une annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<note date="15/04/2004">
<a>Latour</a>
<de>Wolper</de>
</note>
```

- Comme en HTML (`<href="http://...">`), une annotation peut avoir des attributs. Ceux-ci sont toujours compris entre guillemets.

- On a souvent le choix entre inclure une information sous forme d'attribut ou d'élément :

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <note>
  <date>15/04/2004</date>
  ```

- L'élément est préférable quand il s'agit de données ; l'attribut est naturel quand il s'agit de "meta-données" (informations au sujet des données).

Les éléments XML et leur signification

- Les éléments XML peuvent contenir directement de l'information (du texte) ou d'autres éléments.
- Pour que le document soit bien structuré, les annotations d'ouverture et de fermeture doivent se correspondre, ce qui donne au document une structure d'arbre.
- Chaque document comporte un élément de départ appelé élément *racine* (*root element*).
- Les annotations n'ont pas de signification par elles-même, mais peuvent être interprétées par une application qui traite le document.

Les définitions de types de documents

– Un document XML est *bien formé* (*well formed*) s'il est syntaxiquement correct. Il est *valide* (*valid*) s'il est conforme à une *définition de type de document* (*data type definition* - DTD).

– La définition du type de document est mentionnée dans l'entête (ou dans un fichier auxiliaire mentionné dans l'entête).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
  <!ELEMENT note (a,de,sujet,texte)>
  <!ELEMENT a      (#PCDATA)>
  <!ELEMENT de     (#PCDATA)>
  <!ELEMENT sujet  (#PCDATA)>
  <!ELEMENT texte  (#PCDATA)>
]>
<note>
...
</note>
```

Les définitions de types de documents - 2

- La définition d'un type de document définit les éléments qui peuvent apparaître dans le document, ainsi que les annotations correspondantes.

- `<!ELEMENT note (a,de,sujet,texte)>`
  indique que l'élément annoté `note` comporte quatre sous éléments.

- `<!ELEMENT a (#PCDATA)>`
  indique que l'élément annoté `a` est constitué de texte (*Parsed Character Data*). Le "parsed" indique que le texte sera analysé pour y détecter des annotations éventuelles.
  `<!ELEMENT a (#CDATA)>`
  indiquerait que le texte ne doit pas être analysé.

Les définitions de types de documents - 3

– Lorsque l'on décrit la liste des constituants d'un élément, on peut préciser la répétition de certains éléments :

? zéro ou une fois,

* zéro fois ou plus,

+ une fois ou plus.

Exemple :

```
<!ELEMENT note (message+)>
```

– Un choix est indiqué par le symbole |. Exemple :

```
<!ELEMENT note (a,de,header,(message|texte))>
```

Les définitions de types de documents attributs et références

Dans la définition d'un type de document, on peut bien sûr aussi préciser les attributs des différents éléments et leur nature. Exemple :

```
<!ATTLIST note date type CDATA #REQUIRED>
```

- Le dernier élément est la valeur par défaut ; `#REQUIRED` indique qu'il n'y en a pas et qu'une valeur doit figurer dans le document pour l'attribut.
- Différents types sont possibles pour les attributs ; les suivants sont un peu particuliers :

  `ID` un identificateur unique de l'élément,

  `IDREF` une référence à un (identificateur d') élément,

  `IDREFS` une liste de d'identificateurs d'éléments.

  Cela permet de faire référence au même élément à plusieurs endroits d'un document.

Autre méthode de validation : XSD

- XSD = XML Schema Definition

- Exemple :

```
<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="note" type="xs:string" minOccurs="0"/>
      <xs:element name="quantity" type="xs:positiveInteger"/>
      <xs:element name="price" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- A à près équivalent à

```
<!ELEMENT item (title, note?, quantity, price)>
```

Autre méthode de validation : XSD

- Avantages
  - Dialecte XML
  - Permet la spécification plus fine des types de données
  - Autorise la gestion des espaces de nom
  - Plus grande réutilisabilité
  - Globalement plus puissant

- Inconvénients
  - Plus verbeux que DTD
  - Plus complexe à maitriser

Documents XML et Web

- Les navigateurs Web peuvent interpréter le XML.
- Toutefois, sans information additionnelle, ils sont limités à donner la structure du document sans pouvoir interpréter les annotations.
- Une information de mise en page peut être donnée dans des *feuilles de style* (*styles sheets*). Un format utilisé est celui des *cascading style sheets* - *CSS*.
- Une autre possibilité est d'écrire du HTML, mais en se conformant au format XML ; il s'agit du XHTML.

Documents XML avec feuille de style

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="note-style.css"?>
<!DOCTYPE note [
  <!ELEMENT note (a,de,sujet,texte)>
  <!ELEMENT a     (#PCDATA)>
  <!ELEMENT de    (#PCDATA)>
  <!ELEMENT sujet (#PCDATA)>
  <!ELEMENT texte    (#PCDATA)>
]>
<note>
<a>Latour</a>
<de>Wolper</de>
<sujet>Rappel</sujet>
<texte>Ne pas oublier l'examen de BD!</texte>
</note>
```

Exemple de feuille de style

```
note { background-color: #ffffff; width: 100%; }

a{ display: block; color: #00FF00; margin-left: 0;
     font-size: 20pt; }

de { display: block;color: #FF0000; font-size: 20pt; }

sujet { Display: block; color: #0000FF; font-size: 20pt; }

texte { Display: block; color: #000000; margin-top: 20pt;
       margin-left: 20pt; font-size: 20pt; }
```

Autres outils XML

- XPath
  - Langage permettant la sélection des nœuds
- XSLT
  - Permet la transformation d'un document XML en un autre document texte
  - Alternative à XML + CSS
- XQuery
  - Exécution des requêtes sur un (ou plusieurs) document(s) XML
  - Est à XML ce que SQL est aux BDD relationnelles