

Conception des bases de données IV

Bases de données non-relationnelles

bdd4.pdf



STÉPHANE CROZAT

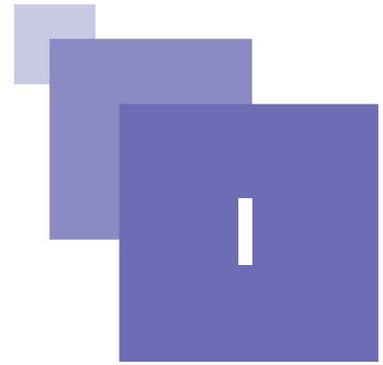
Table des matières

I - Introduction aux bases de données non-relationnelles	6
A. Cours.....	6
1. Perspective technologique et historique : forces et faiblesses du relationnel.....	6
2. Au delà des bases de données relationnelles : Data warehouse, XML et NoSQL.....	7
3. Bases de données NoSQL.....	9
4. Un exemple : Modélisation logique arborescente et objet en JSON.....	13
B. Exercice.....	17
1. Modélisation orientée document avec JSON.....	17
C. Devoir.....	18
1. Document sous licence Creative Commons.....	18
II - Imbrication avec Json et Mongo (base de données orientée document)	20
A. Cours.....	20
1. Exemple de base de données orientée document avec MongoDB.....	20
2. Interroger Mongo en JavaScript.....	23
B. Exercice.....	23
1. Au ciné avec Mongo.....	23
III - Référence d'objets avec Neo4J (BD orientée graphe)	28
A. Cours.....	28
1. Exemple de base de données orientée graphe avec Neo4J.....	28
B. Exercice.....	30
1. De Neo4J à Game of Thrones.....	30
IV - Tables imbriquées en relationnel-objet	36
A. Cours.....	36
1. Introduction aux SGBDRO.....	36
2. Extension du relationnel et du SQL.....	39
3. Les tables imbriquées (nested model).....	42
4. Apport du modèle imbriqué au passage conceptuel-logique.....	45
B. Exercices.....	47
1. MediaTek I.....	47
2. Super-héros relationnels-objets imbriqués.....	47
C. Devoirs.....	48
1. Lab V.....	48
2. Arbre de scène 3D II.....	48

V - Tables imbriquées en relationnel-objet sous Oracle	50
A. Cours.....	50
1. Imbrication en SQL3.....	50
2. Compléments SQL3 pour le modèle imbriqué.....	56
B. Exercices.....	59
1. MediaTek II.....	59
2. RO sans fil imbriqué.....	61
C. Devoirs.....	61
1. Document sous licence Creative Commons.....	61
2. Super-héros relationnels-objets imbriqués, épisode II.....	63
VI - Tables d'objets en relationnel-objet	64
A. Cours.....	64
1. Tables d'objets et identifiants d'objets (OID).....	64
2. Apport des OID au passage conceptuel-logique.....	66
3. Méthodes et héritage dans les tables d'objets.....	69
B. Exercices.....	70
1. MediaTek III.....	70
2. MediaTek IV.....	70
C. Devoirs.....	71
1. Lab VI.....	71
2. Usine de production II.....	72
VII - Tables d'objets en relationnel-objet sous Oracle	73
A. Cours.....	73
1. Tables d'objets et OID en SQL3.....	73
2. Compléments.....	76
B. Exercices.....	79
1. MediaTek V.....	79
2. Des voitures et des hommes.....	81
3. Des voitures et des hommes de collection.....	82
C. Devoirs.....	82
1. Zoologie.....	82
VIII - Modélisation logique arborescente en XML	84
A. Cours.....	84
1. Introduction à XML.....	84
2. Syntaxe de base XML.....	89
3. Introduction aux schémas XML.....	92
4. Manipulation XML avec XPath.....	98
B. Exercice.....	100
1. Mon nom est personne.....	100
2. Glossaire I.....	101
C. Devoir.....	102
1. On l'appelle Trinita.....	102
IX - Introduction aux bases de données XML avec Oracle XMLType	104
A. Cours.....	104
1. Bases de données XML.....	104

B. Exercices.....	109
1. On continue à l'appeler Trinita.....	109
2. T'as le bonjour de Trinita.....	110
C. Devoirs.....	111
1. Documents.....	111
X - Introduction aux data warehouses	112
A. Cours.....	112
1. Data warehouses et bases de données décisionnelles.....	112
2. Pour aller plus loin.....	116
B. Exercice.....	116
1. Étude de cas : Fantastic.....	116
Glossaire	129
Signification des abréviations	130
Bibliographie	131
Webographie	132
Index	133

Introduction aux bases de données non-relationnelles

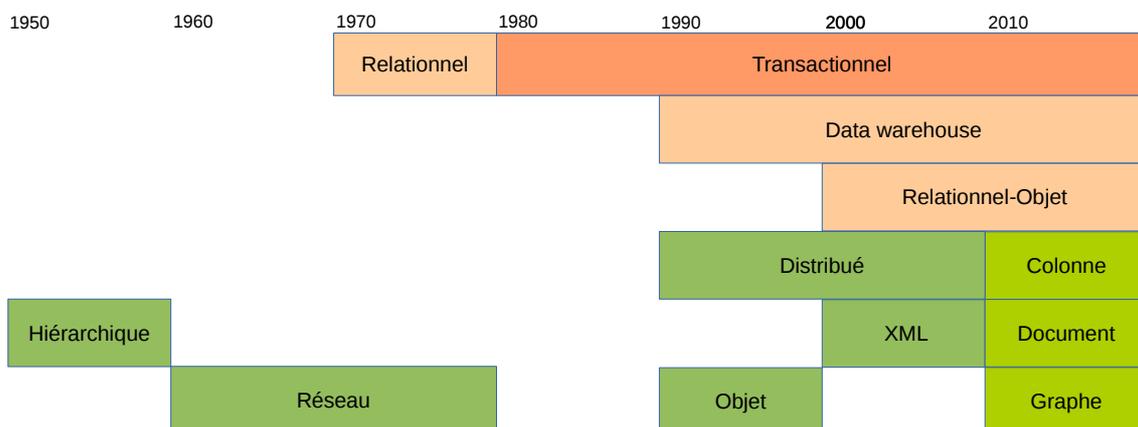


A. Cours

« Un couple de concepteurs de bases de données entre dans un restaurant NoSQL. Une heure après, ils ressortent sans avoir réussi à manger ; ils n'ont jamais réussi à trouver une seule table. »

1. Perspective technologique et historique : forces et faiblesses du relationnel

a) Relationnel et non-relationnel



Les BD NoSQL remettent en cause l'hégémonie des SGBDR telle qu'elle s'est constituée dans les années 1980.

Les BD NoSQL sont essentiellement un retour à des modèles de données antérieurs à cette hégémonie, telles que les représentations hiérarchique ou réseau qui existaient dans les années 60.

b) Domination du relationnel

Fondamental

La première fonction d'une base de données est de permettre de stocker et retrouver l'information.

Relationnel et contrôle de l'intégrité des données

À la naissance de l'informatique, plusieurs modèles de stockage de l'information sont explorés, comme les modèles hiérarchique ou réseau.

Mais c'est finalement le modèle relationnel qui l'emporte dans les années 1970 car c'est lui qui permet de mieux assurer le contrôle de l'intégrité des données, grâce à un modèle théorique puissant et simple.

On notera en particulier :

- Le schéma : on peut exprimer des règles de cohérence a priori et déléguer leur contrôle au système
- La normalisation : on peut supprimer la redondance par un mécanisme de décomposition et retrouver l'information consolidée par les jointures
- La transaction : le système assure le maintien d'états cohérents au sein d'environnements concurrents et susceptibles de pannes

Relationnel et performance en contexte transactionnel

La représentation relationnelle se fonde sur la décomposition de l'information ce qui minimise les entrées/sorties (accès disques, transfert réseau) et permet d'être très performant pour répondre à des questions et des mises à jour ciblées (qui concernent peu de données parmi un ensemble qui peut être très grand). C'est donc une bonne solution dans un contexte transactionnel qui comprend de nombreux accès ciblés à la base.

En revanche ce n'est plus une bonne solution pour des accès globaux à la base (puisqu'il faut alors effectuer beaucoup de jointures pour reconsolider l'ensemble de l'information). C'est le problème posé par le décisionnel.

2. Au delà des bases de données relationnelles : Data warehouse, XML et NoSQL

a) Problème de l'agrégat et développement des data warehouses

Fondamental

Le modèle relationnel est peu performant pour les agrégats.

Problème posé par le décisionnel et résolu par les data warehouses

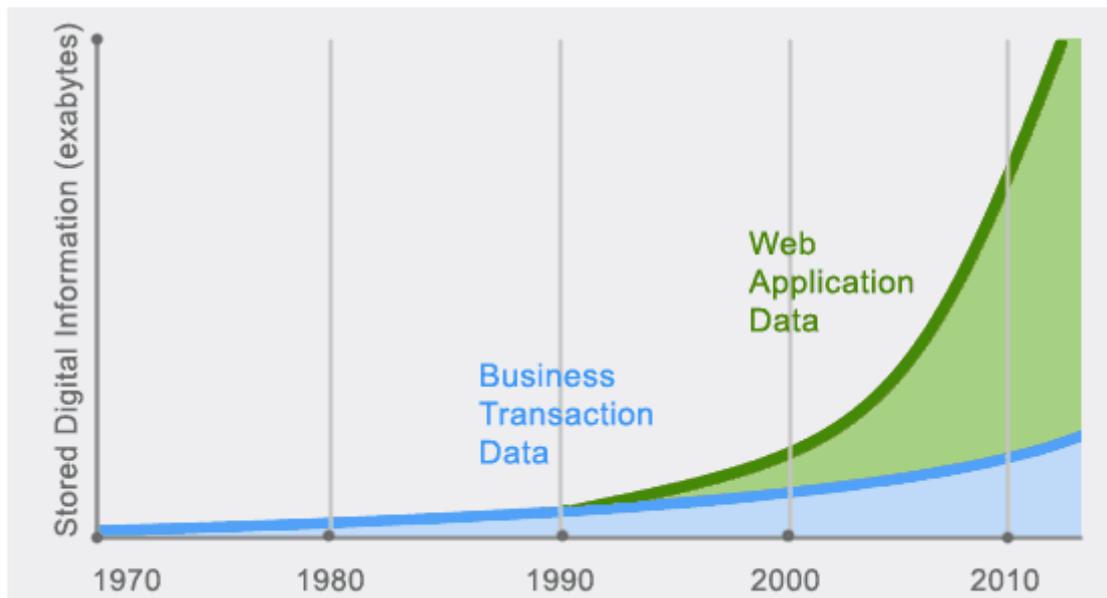
- décision vs gestion
- agrégat vs accès ciblé
- historisation vs transaction

b) Problème de l'impedance mismatch et développement de l'objet et de XML

Fondamental

OID et nested model

c) Problème de la distribution et développement du NoSQL

Big data

Évolution des volumes de données d'entreprise versus web

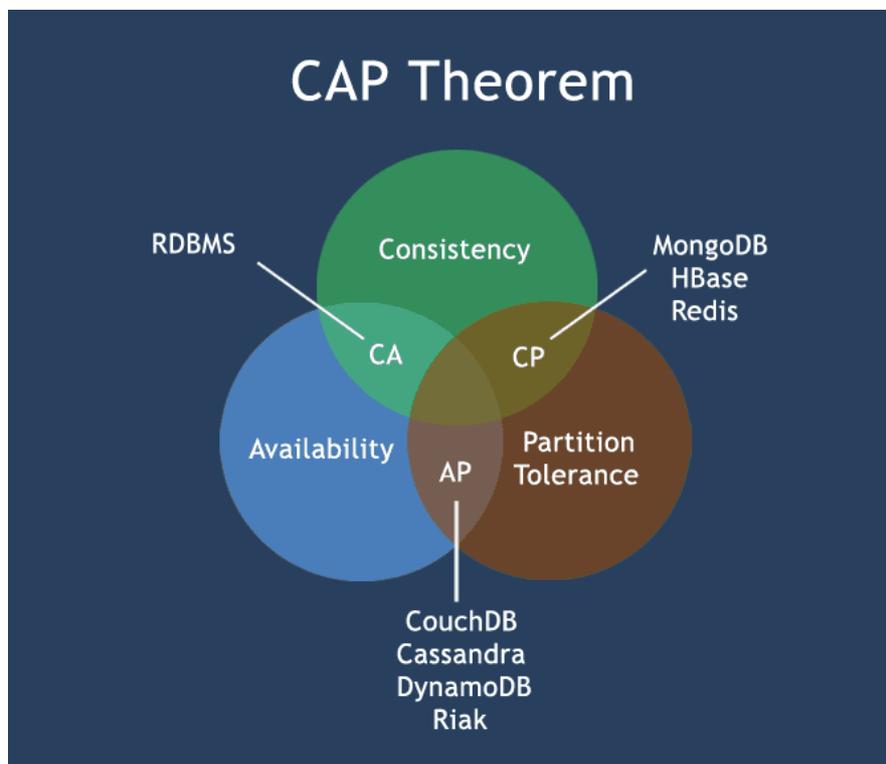
Fondamental : Distribution

Passage du serveur central (*main frame*) à des grappes de machines modestes :

- pour gérer l'explosion des données
- à cause de l'évolution du *hardware*

Fondamental : Le "commerce" de la 3NF

On échange de la performance contre de la souplesse ou contre de la cohérence



Théorème CAP

3. Bases de données NoSQL

a) Définition du mouvement NoSQL

Définition

« Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se caractérisent par une logique de représentation de données non relationnelle et qui n'offrent donc pas une interface de requêtes en SQL.

<http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/>¹



Attention

NoSQL signifie *Not Only SQL* et non pas *No SQL*, il s'agit de compléments aux SGBDR pour des besoins spécifiques et non de solutions de remplacement.

Exemple

- BD orientée clé-valeur
- BD orientée graphe
- BD orientée colonne
- BD orientée document

Complément

<http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/>²

b) Fondamentaux des modèles NoSQL : Clé-valeur, distribution, imbrication, schema-less

Fondamental : Sharding et clé-valeur

- **Simplification du modèle en clé-valeur**
- **Distribution sur les nœuds d'un cluster**

Identification

Object Identifiers (OID)

Universally Unique Identifier (UUID)

Uniform Resource Name (URN)

Imbrication

Structure des valeurs stockées connue par le serveur (RO, JSON, XML, structure interne de type colonne...)

Hachage et distribution

- Logique de dépôt uniquement (stocker et retrouver) ;
- c'est la couche applicative qui fait tout le travail (traitement, cohérence...).

Schema-less

Les bases NoSQL se fondent sur une approche dite *schema-less*, c'est à dire sans schéma logique défini a priori.

1 - <http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/>

2 - <http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/>

L'équivalent du `CREATE TABLE` en SQL n'est soit pas nécessaire, soit même pas possible ; on peut directement faire l'équivalent de `INSERT INTO`.

Cela apporte de la souplesse et de la rapidité, mais se paye avec moins de contrôle et donc de cohérence des données.

Le mouvement NoSQL tend à réintégrer des fonctions de schématisation a priori, à l'instar de ce qui se fait en XML : le schéma est optionnel, mais conseillé en contexte de contrôle de cohérence.

c) Illustration des modèles NoSQL

Exemple : Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Exemple : Représentation de ventes en colonne

Family Sales

#ticket	date	books
1	01/01/16	2212121504 2212141556
2	01/01/16	2212141556

Family Book

#isbn	title	a-surname	a-firstname
2212121504	Scenari		
2212141556	NoSQL	Bruchez	Rudi

*Exemple : Représentation de ventes en document***Collection Sales**

#oid	
4d040766076 6b236450b45 a3	<pre>"ticket" : 1 "date" : "01/01/16" "books" : ["_id" : 2212121504 "_id" : 2212141556]</pre>
4d040766076 6b236450b45 a4	<pre>"ticket" : 2 "date" : "01/01/16" "books" : ["_id" : 2212141556]</pre>

Collection Book

#oid	
4d040766076 6b236450b45 a5	<pre>"isbn" : 2212121504 "title" : "Scenari"</pre>
4d040766076 6b236450b45 a6	<pre>"isbn" : 2212141556 "title" : "NoSQL" "author" : { "surname" : Bruchez "firstname" : Rudi }</pre>

Exemple : Représentation de ventes en document (avec imbrication redondante)

Collection Sales

#oid	
4d040766076 6b236450b45 a3	<pre>"ticket" : 1 "date" : "01/01/16" "books" : [{ "isbn" : 2212121504 "title" : "Scenari" } { "isbn" : 2212141556 "title" : "NoSQL" "author" : { "surname" : Bruchez "firstname" : Rudi } }]</pre>
4d040766076 6b236450b45 a4	<pre>"ticket" : 2 "date" : "01/01/16" "books" : [{ "isbn" : 2212141556 "title" : "NoSQL" "author" : { "surname" : Bruchez "firstname" : Rudi } }]</pre>

Exemple : Représentation de ventes en graphe

Classe Sales

#oid		
4d040766076 6b236450b45 a3	property	ticket : 1
	property	date : 01/01/16
	relation	book : 4d0407660766b236450b45a5
	relation	book : 4d0407660766b236450b45a6
4d040766076 6b236450b45 a4	property	ticket : 2
	property	date : 01/01/16
	relation	book : 4d0407660766b236450b45a6

Classe Book

#oid		
4d040766076 6b236450b45 a5	property	title : Scenari
4d040766076 6b236450b45 a6	property	title : NoSQL
	relation	author : 4d0407660766b236450b45a8

Classe Author

#oid		
4d040766076 6b236450b45 a8	property	surname : Bruchez
	property	firstnam : Rudi

4. Un exemple : Modélisation logique arborescente et objet en JSON

a) JavaScript Object Notation

Définition : Introduction

JSON est un format de représentation logique de données, hérité de la syntaxe de création d'objets en JavaScript.

C'est un format réputé léger (il ne contient pas trop de caractères de structuration), assez facilement lisible par les humains, facilement *parsable* par les machines, et indépendant des langages qui l'utilisent (sa seule fonction est de décrire des données, qui sont ensuite utilisées différemment pour chaque cas suivant le contexte).

Exemple : Un fichier JSON simple

```

1 {
2   "nom" : "Norris",
3   "prenom" : "Chuck",
4   "age" : 73,
5   "etat" : "Oklahoma"
6 }
```

Attention : JSON est Indépendant de tout langage

Bien que JSON puise sa syntaxe du JavaScript, il est indépendant de tout langage de programmation. Il peut ainsi être interprété par tout langage à l'aide d'un *parser*.

Complément : Extension

Un fichier au format JSON a pour extension **".json"**.

b) La syntaxe JSON en bref

Syntaxe : Règles syntaxiques

- Il ne doit exister qu'un seul élément père par document contenant tous les autres : **un élément racine**.
- Tout **fichier JSON bien formé** doit être :
 - **soit un objet** commençant par { et se terminant par },
 - **soit un tableau** commençant par [et terminant par].
 Cependant ils peuvent être vides, ainsi [] et {} sont des JSON valides.
- Les **séparateurs** utilisés entre deux paires/valeurs sont des **virgules**.
- Un objet JSON peut contenir d'autres objets JSON.
- Il ne peut pas y avoir d'éléments croisés.

Fondamental : Éléments du format JSON

Il existe deux types d'éléments :

- **Des couples de type "nom": valeur**, comme l'on peut en trouver dans les **tableaux associatifs**.
- **Des listes de valeurs**, comme les **tableaux utilisés en programmation**.

Définition : Valeurs possibles

- Primitifs : nombre, booléen, chaîne de caractères, null.
- Tableaux : liste de valeurs (tableaux et objets aussi autorisés) entrées entre crochets, séparées par des virgules.
- Objets : listes de couples "nom": valeur (tableaux et objets aussi autorisés) entrés entre accolades, séparés par des virgules.

Exemple

```

1 {
2   "nom cours" : "NF29",
3   "theme" : "ingenierie documentaire",
4   "etudiants" : [
5     {
6       "nom" : "Norris",
7       "prenom" : "Chuck",
8       "age" : 73,
9       "pays" : "USA"
10    },
11    {
12      "nom" : "Doe",
13      "prenom" : "Jane",
14      "age" : 45,
15      "pays" : "Angleterre"
16    },
17    {
18      "nom" : "Ourson",
19      "prenom" : "Winnie",
20      "age" : 10,

```

```

21         "pays" : "France"
22     }
23 ]
24 }

```

c) Principaux usages de JSON

Chargements asynchrones

Avec la montée en flèche des chargements asynchrones tels que l'AJAX (Asynchronous JavaScript And XML) dans le web actuel (qui ne provoquent pas le rechargement total de la page), il est devenu de plus en plus important de pouvoir charger des données organisées, de manière rapide et efficace.

Avec XML, le format JSON s'est montré adapté à ce type de besoins.

Les APIs

Des sociétés telles que Twitter, Facebook ou LinkedIn, offrent essentiellement des services basés sur l'échange d'informations, et font preuve d'un intérêt grandissant envers les moyens possibles pour distribuer ces données à des tiers.

Alors qu'il n'y a pas de domination totale d'un des deux formats (JSON ou XML) dans le domaine des APIs, on constate toutefois que JSON est en train de prendre le pas là où le format XML avait été pionnier.

Exemple : APIs retournant des données au format JSON

Twitter : <https://dev.twitter.com/rest/public>³ : récupération de données du réseau social.

Netatmo : <https://dev.netatmo.com/doc/publicapi>⁴ : récupération de données météo

Les bases de données

Le JSON est très utilisé dans le domaine des bases de données NoSQL (MongoDB, CouchDB, Riak...).

On notera également qu'il est possible de soumettre des requêtes à des SGBDR et de récupérer une réponse en JSON.

Exemple : Fonctions JSON de Postgres et MySQL

Fonctions PostgreSQL : <http://www.postgresql.org/docs/9.3/static/functions-json.html>⁵

Fonctions MySQL : <https://dev.mysql.com/doc/refman/5.7/en/json.html>⁶

d) Exercice

Exercice

Qu'est-ce que le JSON ?

- Un langage de programmation orientée objet
- Un type de pages web
- Un format qui permet de décrire des données structurées
- Une catégorie de documents générés par un traitement de texte

3 - <https://dev.twitter.com/rest/public>

4 - <https://dev.netatmo.com/doc/publicapi>

5 - <http://www.postgresql.org/docs/9.3/static/functions-json.html>

6 - <https://dev.mysql.com/doc/refman/5.7/en/json.html>

Exercice

A quel domaine le JSON n'est-il pas appliqué (à l'heure actuelle) ?

- Le Big Data
- Les chargements asynchrones
- Les APIs
- La mise en forme de pages web

Exercice

Un fichier JSON doit forcément être traité via du code JavaScript.

- Vrai
- Faux

Exercice

On peut utiliser des tableaux en JSON et en XML.

- Vrai
- Faux, on ne peut le faire qu'en XML
- Faux, on ne peut le faire qu'en JSON
- Faux, on ne peut le faire ni en XML ni en JSON

e) Un programme JSON

Soit le fichier HTML et le fichier JavaScript ci-après.

```

1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
4 <title>Démo JSON/JavaScript</title>
5 <script type="text/javascript" src="query.js"></script>
6 </head>
7 <body>
8 <input type="file" id="myfile" onchange="query()"/>
9 <div id="mydiv"/>
10 </body>
11 </html>

```

```

1 function query() {
2 // File
3 var vFile = document.getElementById("myfile").files[0];
4 // Reader
5 var vReader = new FileReader();
6 vReader.readAsText(vFile);
7 vReader.onload = function(pEvent) {
8 // String Input
9 var vContent = pEvent.target.result;
10 // JSON to object
11 var vJson = JSON.parse(vContent);
12 // Query
13 var vResult = vJson.prenom + " " + vJson.nom + " (" + vJson.age + ")";
14 // Output

```

```

15     document.getElementById("mydiv").appendChild(document.createTextNode(vResult));
16 };
17 }

```

Question 1

Écrire un fichier JSON permettant de représenter une personne avec les attributs suivants :

- un nom
- un prénom
- un age
- une liste d'adresses mail

Indice :

La syntaxe JSON

Question 2

Expliquer ce que fait le programme. Tester le programme avec le fichier JSON proposé.

Question 3

Compléter la fonction `query()` afin d'afficher la liste des adresses mail (en plus des information actuelles).

Indice :

On parcourt le tableau JSON à l'aide d'une boucle.

```

1  for (var i=0;i<vJson.adresse.length;i++) {
2      ...
3  }

```

B. Exercice

1. Modélisation orientée document avec JSON

On souhaite réaliser une base de données orientée documents pour gérer des cours et des étudiants, étant données les informations suivantes :

- Un cours est décrit par les attributs code, titre, description, crédits et prérequis.
- Les prérequis sont d'autres cours.
- Un étudiant est décrit par les attributs nom, prénom, adresse.
- Les adresses sont composées d'un numéro de rue, d'une rue, d'une ville et d'un code postal.
- Un étudiant suit plusieurs cours et les cours sont suivis par plusieurs étudiants.

Question 1

Réaliser le MCD en UML de ce problème.

Question 2

Proposer un exemple JSON équivalent à ce que l'on aurait fait en relationnel normalisé (sachant que ce type de solution ne sera généralement pas favorisé en BD orientée document).

Indice :

Représentation d'un cours par un objet JSON.

```
1 {  
2   "code": "api04",  
3   "titre": "DW et NOSQL"  
4   ...  
5 }
```

Question 3

Proposer un exemple JSON basé sur l'imbrication.

Quel est le principal défaut de cette solution ?

Est-il toujours possible d'avoir une solution ne reposant que sur l'imbrication ?

Question 4

Proposer un exemple JSON basé sur les références.

Quelles sont les principales différences avec un système relationnel ?

Question 5

Sachant que l'objectif de l'application est de visualiser une liste des étudiants avec les cours que chacun suit, et d'accéder aux détails des cours uniquement lorsque l'on clique sur son code ou son titre.

Proposer une solution adaptée à ce problème mobilisant référence et imbrication.

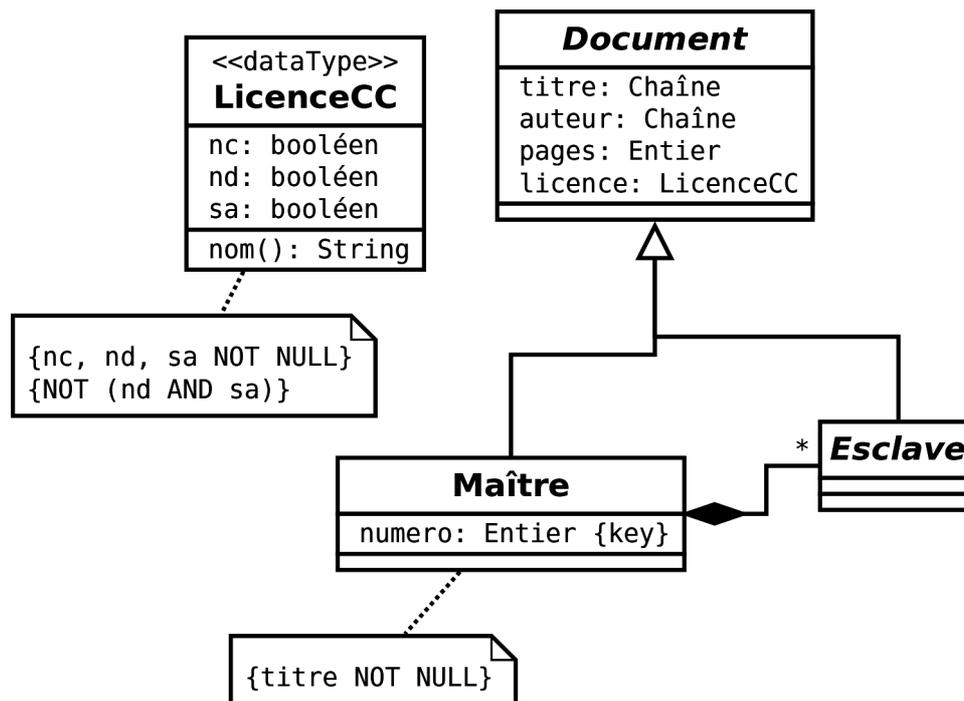
C. Devoir

1. Document sous licence Creative Commons

On souhaite créer un langage permettant de d'assembler des documents situés dans une base de données. Un document est décrit par un titre, un auteur, un nombre de pages et une licence *creative commons* (CC) : cc-by, cc-by-nc, cc-by-nd, cc-by-sa, cc-by-nc-nd, cc-by-nc-sa.

Un document peut être :

- soit un document maître, il dispose alors toujours d'un numéro d'enregistrement unique dans la base, et son titre sera toujours renseigné ;
- soit un document esclave, il est alors membre d'un document maître.



Question 1

Proposez une représentation JSON des données (le document numéro 1 et ses deux documents esclaves) en vue d'expérimenter une solution MongoDB.

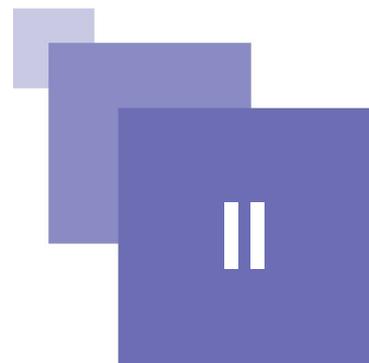
Indice :

On notera que le numéro d'enregistrement doit être intégré au JSON, c'est une clé naturelle, puisqu'il fait partie du cahier des charges (il n'est pas ajouté au moment de l'implémentation technique, mais dès la modélisation UML).

Question 2

Donner un avantage et un inconvénient de cette approche non-relationnelle par rapport à une approche relationnelle. Vous n'avez le droit qu'à une seule phrase précise et synthétique pour l'avantage et une autre pour l'inconvénient, chacune s'appuyant sur le cas précis de l'exercice.

Imbrication avec Json et Mongo (base de données orientée document)



A. Cours

1. Exemple de base de données orientée document avec MongoDB

a) Présentation de MongoDB

MongoDB est une base de données *open source* NoSQL orientée document. Elle stocke des données au format JSON (en fait BSON, qui est une version binaire de JSON).

Le serveur MongoDB est organisé en plusieurs *databases* :

- Chaque *database* contient des *collections*.
- Chaque *collection* contient des *documents*.
- Chaque *document* est au format JSON et contient donc des propriétés.

SQL	MongoDB
base de données et/ou schéma	base de données
table	collection
enregistrement	document
attribut (atomique)	propriété (chaîne, entier, tableau, structure)

Tableau 1 Comparaison SQL / MongoDB

Schema-less

C'est une base *schema-less*, aussi une collection peut contenir des documents de structures différentes et il n'est pas possible de définir la structure a priori d'une collection. La structure

d'une collection n'est donc définie que par les document qui la compose, et elle peut évoluer dynamiquement au fur et à mesure des insertions et suppressions.

Identification clé / valeur

Chaque document est identifié par un identifiant nommé `_id` unique pour une collection, fonctionnant comme une **clé primaire artificielle**.

Architecture

MongoDB fonctionne a minima sous la forme d'un serveur auquel il est possible de se connecter avec un client textuel (*mongo shell*).

MongoDB peut être distribuée sur plusieurs serveurs (partitionnement horizontal ou *sharding*) et accédée à travers de multiples couches applicatives (langages, API...)

Complément

<https://docs.mongodb.org/manual/>⁷

b) Installation et démarrage de MongoDB

MongoDB est disponible sur Windows, OSX et Linux.

<https://docs.mongodb.com/manual/installation/>⁸

Client textuel Mongo (mongo shell)

Pour se connecter à un serveur MongoDB :

- présent sur la même machine : exécuter simplement `mongo` dans un terminal ;
- distant sur le port standard de MongoDB : exécuter `mongo --host nom-du-serveur`.

Test

Une fois connecté exécuter le code suivant pour vérifier le bon fonctionnement de la base :

```
1 db.test.insert({ "a":0 })
2 db.test.find()
```

Le résultat attendu est le suivant, la clé générée étant bien entendu différente :

```
{ "_id":ObjectId("574ebe32b3286a9a8fadfb55"), "a":0 }
```

Complément

<https://docs.mongodb.com/manual/mongo/>⁹

c) Créer des bases de données et des collections

Fondamental : Créer une base et une collection

MongoDB est une base *schema-less*, la création des bases et des collections est dynamique lors d'une première **insertion** de données.

Méthode

Pour créer une base de données il faut exécuter une instruction `use` sur une nouvelle base de données, puis donner un ordre d'insertion d'un premier document JSON avec `insert`.

Exemple

```
use db1
```

7 - <https://docs.mongodb.org/manual/>

8 - <https://docs.mongodb.com/manual/installation/>

9 - <https://docs.mongodb.com/manual/mongo/>

```
db.coll.insert( { "x":1 } )
```

Syntaxe : Catalogue de données

- On peut voir la liste des bases de données avec :
`show dbs`
- On peut voir la liste des collections de la base de données en cours avec :
`show collections`
- On peut voir le contenu d'une collection avec :
`db.coll.find()`

Complément

<https://docs.mongodb.com/manual/mongo>

d) Insertion des documents

L'insertion de données dans une base MongoDB se fait avec l'instruction `db.collection.insert(Document JSON)`.

Si la collection n'existe pas, elle est créée dynamiquement.

L'insertion associe un identifiant (`_id`) à chaque document de la collection.

Exemple

```

1 db.Cinema.insert(
2 {
3   "nom": "Honkytonk Man",
4   "realisateur": {
5     "nom": "Eastwood",
6     "prenom": "Clint"
7   },
8   "annee": 1982,
9   "acteurs": [
10    {
11      "nom": "Eastwood",
12      "prenom": "Kyle"
13    },
14    {
15      "nom": "Eastwood",
16      "prenom": "Clint"
17    }
18  ]
19 }
20 )

```

Complément

<https://docs.mongodb.org/manual/core/crud>¹⁰

<https://docs.mongodb.com/manual/tutorial/insert-documents>¹¹

e) Trouver des documents

La recherche de données dans une base MongoDB se fait avec l'instruction `db.collection.find(Document JSON, document JSON)`, avec :

- le premier document JSON définit une restriction ;
- le second document JSON définit une projection (ce second argument est optionnel).

10 - <https://docs.mongodb.org/manual/core/crud/>

11 - <https://docs.mongodb.com/manual/tutorial/insert-documents/>

Exemple : Restriction

```
1 db.Cinema.find({"nom":"Honkytonk Man"})
```

retourne les document JSON tels qu'ils ont à la racine un attribut "nom" avec la valeur "Honkytonk Man".

Exemple : Restriction et projection

```
1 db.Cinema.find({"nom":"Honkytonk Man"}, {"nom":1, "realisateur":1} )
```

retourne les document JSON tels qu'ils ont à la racine un attribut "nom" avec la valeur "Honkytonk Man", et seul les attributs situés à la racine "nom" et "realisateur" sont projetés (en plus de l'attribut "_id" qui est projeté par défaut).

Complément

<https://docs.mongodb.org/manual/tutorial/query-documents/>¹²

2. Interroger Mongo en JavaScript

a) Interroger Mongo en JavaScript

Le console mongo permet d'exécuter des programme JavaScript avec instruction load.

```
1 //test.js
2 print("Hello world");
```

```
1 > load("test.js")
```

Parcours d'un résultat de requête Mongo

```
1 //query.js
2 conn = new Mongo();
3 db = conn.getDB("db1");
4
5 recordset = db.User.find({"liked":{"$elemMatch":{"star":3}}}, {"_id":0,
6 "liked.film":1})
7
8 while ( recordset.hasNext() ) {
9     printjson( recordset.next() );
10 }
```

Complément

<https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/>¹³

12 - <https://docs.mongodb.org/manual/tutorial/query-documents/>

13 - <https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/>

B. Exercice

1. Au ciné avec Mongo

[1h30]

Soit les données suivantes représentant des films de cinéma.

```
1 db.Cinema.drop()
2
3 db.Cinema.insert(
4 {
5   nom:"Goodfellas",
6   annee:1990,
7   realisateur:{nom:"Scorsese", prenom:"Martin"},
8   acteurs:
9     [
10    {nom:"De Niro", prenom:"Robert"},
11    {nom:"Liotta", prenom:"Ray"},
12    {nom:"Pesci", prenom:"Joe"}
13   ]
14 })
15
16 db.Cinema.insert(
17 {
18   nom:"The Godfather",
19   annee:1972,
20   realisateur:{nom:"Coppola", prenom:"Francis Ford"},
21   acteurs:
22     [
23     {nom:"Pacino", prenom:"Al"},
24     {nom:"Brando", prenom:"Marlon"},
25     {nom:"Duvall", prenom:"Robert"}
26     ]
27 })
28
29 db.Cinema.insert(
30 {
31   nom:"Million Dollar Baby",
32   realisateur:{nom:"Eastwood", prenom:"Clint"},
33   acteurs:
34     [
35     {nom:"Swank", prenom:"Hilary"},
36     {nom:"Eastwood", prenom:"Clint"}
37     ]
38 })
39
40 db.Cinema.insert(
41 {
42   nom:"Gran Torino",
43   annee:2008,
44   realisateur:{nom:"Eastwood", prenom:"Clint"},
45   acteurs:
46     [
47     {nom:"Vang", prenom:"Bee"},
48     {nom:"Eastwood", prenom:"Clint"}
49     ]
50 })
51
52 db.Cinema.insert(
53 {
54   nom:"Unforgiven",
55   realisateur:{nom:"Eastwood", prenom:"Clint"},
56   acteurs:
57   [
```

```

58   {nom:"Hackman", prenom:"Gene"},
59   {nom:"Eastwood", prenom:"Clint"}
60 ]
61 })
62
63 db.Cinema.insert(
64 {
65   nom:"Mystic River",
66   realiseur:{nom:"Eastwood", prenom:"Clint"},
67   acteurs:
68   [
69     {nom:"Penn", prenom:"Sean"},
70     {nom:"Bacon", prenom:"Kevin"}
71   ]
72 })
73
74 db.Cinema.insert(
75 {
76   nom:"Honkytonk Man",
77   realiseur:{nom:"Eastwood", prenom:"Clint"},
78   annee:1982,
79   acteurs:
80   [
81     {nom:"Eastwood", prenom:"Kyle"},
82     {nom:"Bloom", prenom:"Verna"}
83   ]
84 })
85
86 db.Cinema.find()

```

L'objectif est d'initialiser une base MongoDB avec ce script, puis d'écrire les requêtes MongoDB permettant de répondre aux questions suivantes.

Question 1

Créer une nouvelle base MongoDB et exécuter le script. Nommez votre base par votre nom de famille ou votre login sur la machine par exemple.

Indices :

Pour créer une base de données, utiliser l'instruction `use myNewDatabase`, puis exécuter au moins une instruction d'insertion.

Créer des bases de données et des collections

Question 2

Quels sont les films sortis en 1990 ?

Indices :

Trouver des documents

`db.Cinema.find(document JSON)`

La syntaxe JSON

`db.col.find({"attribute":"value"})`

Question 3

Quels sont les films sortis avant 2000 ?

Indice :

On utilisera l'objet `{$lt:value}` à la place de la valeur de l'attribut à tester (`$lt` pour lesser than).

Question 4

Quels sont les films réalisés par Clint Eastwood ?

Indice :

On utilisera un objet comme valeur.

Question 5

Quels sont les films réalisés par quelqu'un prénommé Clint ?

Indice :

Utiliser le navigateur de propriété des objets point : `object.attribute`.

Question 6

Quels sont les films réalisés par quelqu'un prénommé Clint avant 2000 ?

Indice :

Utiliser une liste de conditions `attribut:valeur` pour spécifier un AND (et logique) :
`db.col.find({"attribute1":"value1", "attribute2":"value2"})`

Question 7

Quels sont les films dans lesquels joue Clint Eastwood ?

Question 8

Quels sont les films dans lesquels joue un Eastwood ?

Indice :

Utiliser le parser de tableau `$elemMatch: {"key": "value"}` à la place de la valeur.

Question 9

Quels sont les noms des films dans lesquels joue un Eastwood ?

Indices :

*Pour gérer la **projection**, utiliser un second argument de la clause `find()` :*
`db.Cinema.find({document JSON de sélection }, {document JSON de projection})`
avec document JSON de projection de la forme : `{"attribut1":1, "attribut2":1...}`
Les identifiants sont toujours affichés par défaut, si on veut les supprimer, on peut ajouter la clause `_id:0` dans le document de projection.

Question 10

Compléter le programme JavaScript suivant afin d'afficher les titre selon le format suivant :

```
1 - Million Dollar Baby
2 - Gran Torino
3 - Unforgiven
4 - Honkytonk Man
```

Indice :

```
1 conn = new Mongo();
2 db = conn.getDB("...");
3
4 recordset = ...
5
6 while ( recordset.hasNext() ) {
7     film = recordset.next() ;
8     print("- ", ...);
```

```
9 } }
```

On veut à présent ajouter une nouvelle collection permettant de gérer des utilisateurs et leurs préférences. Pour chaque utilisateur on gèrera un pseudonyme, et une liste de films préférés avec une note allant de une à trois étoiles.

Question 11

Ajouter trois utilisateurs préférant chacun un ou deux films.

On utilisera les identifiants des films pour les référencer.

Indice :

Insertion des documents

Question 12

Critiquer cette insertion mobilisant les identifiants des films ? Pourquoi n'est ce pas reproductible ?

Question 13

Quels sont les utilisateurs qui aiment au moins un film avec 3 étoiles ?

Question 14

Quels sont les identifiants des films qui sont aimés au moins une fois avec 3 étoiles ?

Question 15

Pourquoi trouver les titres de ces films n'est pas simple avec Mongo ?

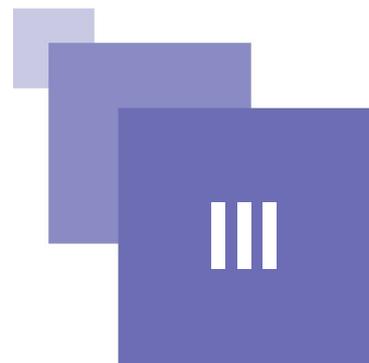
Question 16

Écrire le programme JavaScript permettant de récupérer la liste dédoublonnée des identifiants de films qui sont aimés au moins une fois avec 3 étoiles, puis d'afficher les titres des films correspondants.

Indice :

```
1 conn = new Mongo();
2 db = conn.getDB("...");
3
4 films = ...
5
6 for (var i=0; i<films.length; i++) {
7   var id = films[i];
8   if (film = ...)
9     print(...);
10 }
11
```

Référence d'objets avec Neo4J (BD orientée graphe)



A. Cours

1. Exemple de base de données orientée graphe avec Neo4J

a) Installation de Neo4J

L'installation présentée ici est uniquement destinée à tester Neo4J localement. L'installation de Neo4j peut être réalisée localement sans droit administrateurs.

Installation de Neo4J

1. Télécharger depuis : <http://neo4j.com/download/>¹⁴
2. Choisir la version `Community Edition`
3. Suivez la procédure suivant votre système d'exploitation (Mac OSX, Linux/UNIX, Windows)

Complément : Pré-requis Java 8

<http://neo4j.com/docs/stable/deployment-requirements.html>¹⁵

b) Démarrage de Neo4J

Lancer le serveur

Le démarrage du serveur dépend du système d'exploitation.

- Sous Windows et Mac OSX lancer l'application et cliquer sur le bouton `Start`
- Sous Linux exécuter : `$NEO4J_HOME/bin/neo4j console`

Lancer le client

Ouvrir un navigateur Web à l'adresse : <http://localhost:7474>¹⁶

Complément

- Pour arrêter le processus serveur sous Linux : `$NEO4J_HOME/bin/neo4j stop`

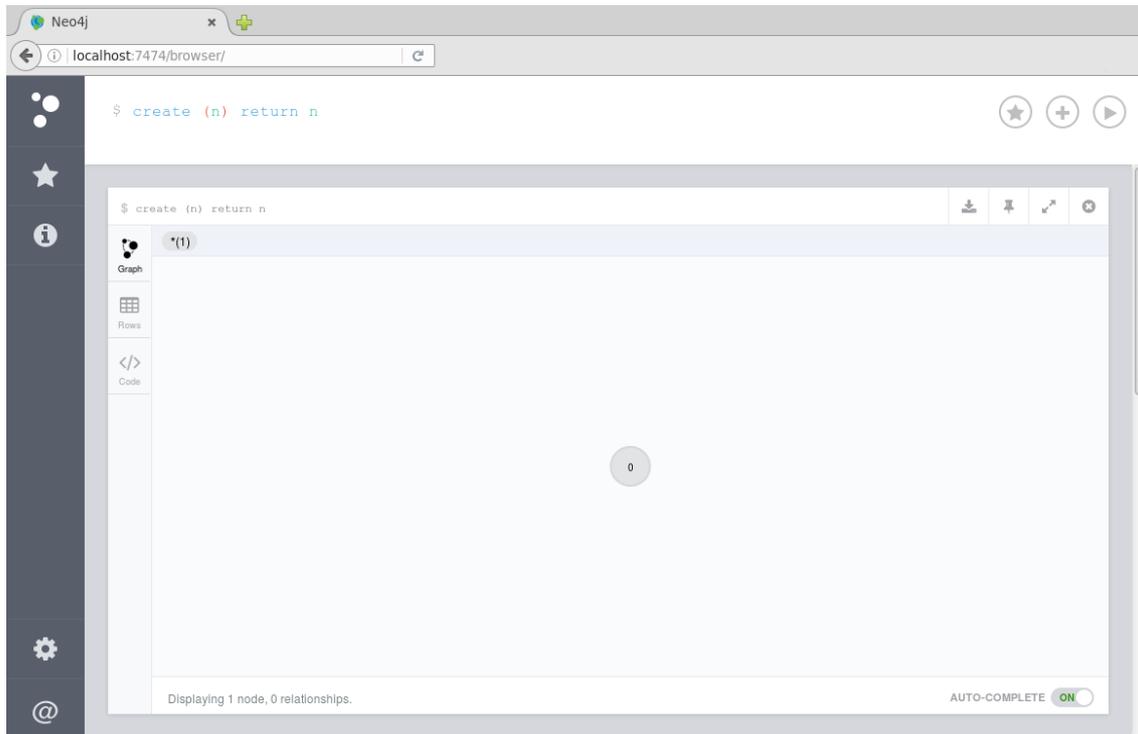
14 - <http://neo4j.com/download/>

15 - <http://neo4j.com/docs/stable/deployment-requirements.html>

16 - <http://localhost:7474>

- Pour réinitialiser les bases de données : supprimer le dossier \$NEO4J_HOME/data/graph.db/

c) Test de Neo4J



Interface de Neo4j (création d'un nœud)

- Les requêtes s'écrivent dans la partie haute.
- Les résultats se visualisent dans la partie basse.

Exemple : Créer un nœud

```
create (n) return n
```

Exemple : Voir tous les nœuds

```
match (n) return n
```

d) Créer des nœuds et des relations

Exemple : Créer un nœud avec une propriété

```
create (n {name:'Sun'}) return n
```

Exemple : Créer un nœud avec un label

```
create (n:Astre {name:'Sun'}) return n
```

Syntaxe : Créer des relations

le formalisme pour les relations est :

- ()<-[]-()
- ()-[]->()
- ()<-[]->()
- ()-[]-()

où :

- on mettra nos nœuds à l'intérieur des parenthèses ()
- on mettra nos relations à l'intérieur des crochets []
- enfin on désignera le sens de la relation avec la flèche

Exemple : Créer deux nœuds et une relation entre ces deux nœuds

```
create (e:Astre {name:'Earth'}) <-[:Satellite {distance:384000}]- (m:Astre {name:'Moon'}) return e, m
```

Exemple : Créer deux nœuds et une relation entre ces deux nœuds en plusieurs instructions

```
create (e:Astre {name:'Earth'})
create (m:Astre {name:'Moon'})
create (e)<-[:Satellite {distance:384000}]- (m)
return e, m
```

Remarque

- CTRL+ENTER
Lance la commande présente dans la ligne de commande.
- SHIFT+ENTER
Permet de faire un saut de ligne dans la ligne de commande (sans l'exécuter)

e) Trouver des nœuds et des relations

Exemple : Trouver un nœud avec une propriété

```
match (n {name:'Sun'}) return n
```

Exemple : Trouver un nœud avec un label

```
match (n:Astre) return n
```

Exemple : Trouver des nœuds en fonction d'une relation

```
match (n1) <-[:Satellite]- (n2) return r,n1,n2
```

B. Exercice

1. De Neo4J à Game of Thrones

[1h]

Un beau matin, vous vous réveillez en Westeros, le monde de Game Of Thrones, sans aucune connaissance historique. Afin d'y survivre, il va falloir intégrer les informations du royaume. Nous allons pour cela utiliser une base de données orientée graphe Neo4J.



Question 1

Avant toute chose vous allez vous allez devoir vous ajouter au monde que vous venez de rejoindre.

Créer un nouveau personnage avec votre nom et prénom.

Indices :

Créer des nœuds et des relations

Le label est :personnage.

Les propriétés sont :

```
{name : '...', nickname : '...' }
```

Question 2

Seconde chose, il faut apprendre à créer des relations, car le monde de Game Of Thrones est très complexe.

Créez deux personnages et exprimez une relation qui les relie.

Par exemple, un des combats les plus célèbres oppose Oberyn Martell, dit The Viper et Gregor Clegane, dit The Mountain. Gregor Clegane tue Oberyn Martell en duel.

Indices :

Créer des nœuds et des relations

Créer d'abord les deux nœuds avec l'instruction CREATE.

```
1 CREATE(gregor:personnage { name : '...', nickname : '...' })
2 CREATE(oberyn:...)
3 ...
```

Toujours avec CREATE, créer une relation de type « :TUE » entre les deux nœuds avec comme attribut : « type : 'duel' »

```
1 CREATE(gregor:personnage { name : 'Gregor Clegane', nickname : 'The Mountain' })
```

```

2 CREATE(oberyn:personnage { name : 'Oberyn Martell', nickname : 'The
  Viper' })
3 CREATE(...)-[...]->(...)
4 ...

```

```

1 CREATE(gregor:personnage { name : 'Gregor Clegane', nickname : 'The
  Mountain' })
2 CREATE(oberyn:personnage { name : 'Oberyn Martell', nickname : 'The
  Viper' })
3 CREATE(gregor)-[...]->(oberyn)
4 RETURN gregor, oberyn

```

```

1 CREATE(gregor:personnage { name : 'Gregor Clegane', nickname : 'The
  Mountain' })
2 CREATE(oberyn:personnage { name : 'Oberyn Martell', nickname : 'The
  Viper' })
3 CREATE(gregor)-[:TUE { type : ... }]->(oberyn)
4 RETURN gregor, oberyn

```

Question 3

Affichez l'ensemble des nœuds présents dans votre base de données.

Indice :

Test de Neo4J

Nous pouvons à présent alimenter notre base d'un ensemble de connaissance dont nous disposons. Pour cela copier et exécuter le code Cypher ci-après.

```

1 //GameOfThrones:
2
3 //sans clan :
4 CREATE (sansclan:clan { clannome : 'Clan des sans lien de sang'})
5
6 CREATE (hodor:personnage { name : 'Hodor', nickname : 'I am the only good actor
  here !' })
7 CREATE (khal:personnage { name : 'Khal Drogo', nickname : 'Horse Man' })
8 CREATE (petyr:personnage { name : 'Petyr Baelish', nickname : 'LittleFinger' })
9 CREATE (shae:personnage { name : 'Shae for the Imp', nickname : 'Dead' })
10
11
12 //lanister :
13 CREATE (lanister:clan { clannome : 'The Lannisters'})
14
15 CREATE (tyrion:personnage { name : 'Tyrion Lannister', nickname : 'The Imp' })
16 CREATE (tywin:personnage { name : 'Tywin Lannister', nickname : 'Father of the
  Lannisters' })
17 CREATE (jaime:personnage { name : 'Jaime Lannister', nickname : 'King Slayer' })
18 CREATE (cersei:personnage { name : 'Cersei Lannister', nickname : 'Brother Lover'
  })
19 CREATE (joffrey:personnage { name : 'Joffrey Lannister', nickname : 'Mad King
  2' })
20
21
22 //baratheons :
23 CREATE (baratheon:clan { clannome : 'The Baratheons'})
24
25 CREATE (robert:personnage { name : 'Robert Baratheon', nickname : 'King
  Robert' })
26 CREATE (renly:personnage { name : 'Renly Baratheon', nickname : 'King Gay' })
27 CREATE (stannis:personnage { name : 'Stannis Baratheon', nickname : 'Stéphane' })
28 CREATE (gendry:personnage { name : 'Gendry ???Baratheon', nickname : 'What
  happened to my story ?' })
29
30

```

```

31 //tyrells :
32 CREATE (tyrell:clan { clanname : 'The Tyrells'})
33
34 CREATE (margeary:personnage { name : 'Margeary Tyrell', nickname : 'Tyrell asset'
35 })
36 CREATE (loras:personnage { name : 'Loras Tyrell', nickname : 'King Gay Knight' })
37
38 //targaryens :
39 CREATE (targaryen:clan { clanname : 'The Targaryen'})
40
41 CREATE (daenerys:personnage { name : 'Daenerys Targaryen', nickname : 'Mother of
42 Dragons' })
43 CREATE (viserys:personnage { name : 'Viserys Targaryen', nickname : 'Gold
44 Head' })
45
46 //Stark :
47 CREATE (stark:clan { clanname : 'The Starks'})
48
49 CREATE (arya:personnage { name : 'Arya Stark', nickname : 'I am 20' })
50 CREATE (sansa:personnage { name : 'Sansa Stark', nickname : 'I am sorry I can t
51 smile.' })
52 CREATE (rosse:personnage { name : 'Roose Bolton', nickname : 'Come by the house I
53 ll kill you' })
54 CREATE (edward:personnage { name : 'Eddard Stark', nickname : 'Whhhhhy ???' })
55 CREATE (robb:personnage { name : 'Robb Stark', nickname : 'King of the North' })
56 CREATE (john:personnage { name : 'John Snow', nickname : 'The Bastard' })
57 CREATE (bran:personnage { name : 'Bran Stark', nickname : 'Stop boring me' })
58
59 //tullys :
60 CREATE (tully:clan { clanname : 'The Tullys'})
61
62 CREATE (catelyn:personnage { name : 'Catelyn Stark', nickname : 'Mother of
63 Stark ?' })
64 CREATE (lysa:personnage { name : 'Lyse Tully', nickname : 'Crazy' })
65
66 //greyjoys :
67 CREATE (greyjoy:clan { clanname : 'The Greyjoys'})
68
69 CREATE (theon:personnage { name : 'Theon Greyjoy', nickname : 'The enuque' })
70 CREATE (balon:personnage { name : 'Balon Greyjoy', nickname : 'Nicest father
71 ever' })
72
73 //actions :
74
75 CREATE (tyrion)-[:TUE{type : 'homicide'}]->(shae)
76 CREATE (tyrion)-[:TUE{type : 'homicide'}]->(tywin)
77 CREATE (petyr)-[:TUE{type : 'homicide'}]->(joffrey)
78 CREATE (stannis)-[:TUE{type : 'homicide'}]->(renly)
79 CREATE (khal)-[:TUE{type : 'homicide'}]->(khal)
80 CREATE (khal)-[:TUE{type : 'homicide'}]->(viserys)
81 CREATE (joffrey)-[:TUE{type : 'homicide'}]->(edward)
82 CREATE (rosse)-[:TUE{type : 'homicide'}]->(robb)
83 CREATE (rosse)-[:TUE{type : 'homicide'}]->(catelyn)
84 CREATE (petyr)-[:TUE{type : 'homicide'}]->(lysa)
85
86 CREATE (jaime)-[:AIME{type : 'amour'}]->(cersei)
87 CREATE (cersei)-[:AIME{type : 'amour'}]->(jaime)
88 CREATE (tyrion)-[:AIME{type : 'amour'}]->(shae)
89 CREATE (shae)-[:AIME{type : 'amour'}]->(tywin)
90 CREATE (robert)-[:AIME{type : 'amour'}]->(cersei)
91 CREATE (loras)-[:AIME{type : 'amour'}]->(renly)
92 CREATE (renly)-[:AIME{type : 'amour'}]->(loras)
93 CREATE (margeary)-[:AIME{type : 'amour'}]->(joffrey)
94 CREATE (joffrey)-[:AIME{type : 'amour'}]->(margeary)

```

```

93 CREATE (khal)-[:AIME{type : 'amour'}]->(daenerys)
94 CREATE (daenerys)-[:AIME{type : 'amour'}]->(khal)
95 CREATE (petyr)-[:AIME{type : 'amour'}]->(catelyn)
96 CREATE (edward)-[:AIME{type : 'amour'}]->(catelyn)
97 CREATE (catelyn)-[:AIME{type : 'amour'}]->(edward)
98
99
100 //liens de clan :
101
102 CREATE (theon)-[:LIER{type : 'liendeclan'}]->(greyjoy)
103 CREATE (balon)-[:LIER{type : 'liendeclan'}]->(greyjoy)
104
105 CREATE (khal)-[:LIER{type : 'liendeclan'}]->(sansclan)
106 CREATE (john)-[:LIER{type : 'liendeclan'}]->(sansclan)
107 CREATE (petyr)-[:LIER{type : 'liendeclan'}]->(sansclan)
108 CREATE (gendry)-[:LIER{type : 'liendeclan'}]->(sansclan)
109 CREATE (hodor)-[:LIER{type : 'liendeclan'}]->(sansclan)
110
111 CREATE (gendry)-[:LIER{type : 'liendeclan'}]->(baratheon)
112 CREATE (joffrey)-[:LIER{type : 'liendeclan'}]->(baratheon)
113 CREATE (robert)-[:LIER{type : 'liendeclan'}]->(baratheon)
114 CREATE (renly)-[:LIER{type : 'liendeclan'}]->(baratheon)
115 CREATE (stannis)-[:LIER{type : 'liendeclan'}]->(baratheon)
116
117 CREATE (margary)-[:LIER{type : 'liendeclan'}]->(tyrell)
118 CREATE (loras)-[:LIER{type : 'liendeclan'}]->(tyrell)
119
120 CREATE (daenerys)-[:LIER{type : 'liendeclan'}]->(targaryen)
121 CREATE (viserys)-[:LIER{type : 'liendeclan'}]->(targaryen)
122
123 CREATE (lysa)-[:LIER{type : 'liendeclan'}]->(tully)
124 CREATE (catelyn)-[:LIER{type : 'liendeclan'}]->(tully)
125
126 CREATE (arya)-[:LIER{type : 'liendeclan'}]->(stark)
127 CREATE (hodor)-[:LIER{type : 'liendeclan'}]->(stark)
128 CREATE (rosse)-[:LIER{type : 'liendeclan'}]->(stark)
129 CREATE (sansa)-[:LIER{type : 'liendeclan'}]->(stark)
130 CREATE (petyr)-[:LIER{type : 'liendeclan'}]->(stark)
131 CREATE (edward)-[:LIER{type : 'liendeclan'}]->(stark)
132 CREATE (robb)-[:LIER{type : 'liendeclan'}]->(stark)
133 CREATE (john)-[:LIER{type : 'liendeclan'}]->(stark)
134 CREATE (bran)-[:LIER{type : 'liendeclan'}]->(stark)
135 CREATE (catelyn)-[:LIER{type : 'liendeclan'}]->(stark)
136 CREATE (theon)-[:LIER{type : 'liendeclan'}]->(stark)
137
138 CREATE (shae)-[:LIER{type : 'liendeclan'}]->(lanister)
139 CREATE (rosse)-[:LIER{type : 'liendeclan'}]->(lanister)
140 CREATE (tyrion)-[:LIER{type : 'liendeclan'}]->(lanister)
141 CREATE (tywin)-[:LIER{type : 'liendeclan'}]->(lanister)
142 CREATE (jaime)-[:LIER{type : 'liendeclan'}]->(lanister)
143 CREATE (cersei)-[:LIER{type : 'liendeclan'}]->(lanister)
144 CREATE (joffrey)-[:LIER{type : 'liendeclan'}]->(lanister)

```

Question 4

Affichez l'ensemble des nœuds présents dans votre base de données.
Manipulez le graphe quelques minutes afin de vous faire une idée des données.

Question 5

Affichez à présent la liste des clans.

Indice :

*On utilise le label :clan après l'alias de nœud n.
Trouver des nœuds et des relations*

Question 6

Afficher le nœud qui possède comme *nickname* The Imp.

Question 7

Affichez le clan qui a pour *clanname* The Starks.

Question 8

Affichez tous les personnages qui ont une relation *:LIER* avec le clan The Starks.

Indices :

```
1 match (c:clan {clanname:'The Starks'})<-[...]-(...) return ...
```

```
1 match (c:clan {clanname:'The Starks'})<-[l:...]->(p) return l,c,p
```

Question 9

Affichez toutes les relations de type *:TUE*, pour savoir qui a tué qui.

Indice :

```
1 match (:personnage)-[...]->(:personnage)
2 return result
```

Tables imbriquées en relationnel-objet

IV

A. Cours

Nous étudions dans ce module le modèle imbriqué, c'est à dire l'extension relationnel-objet qui permet d'insérer des objets sous forme de vecteurs, ou des collections d'objets sous forme de tables dans un attribut d'une table.

1. Introduction aux SGBDRO

Objectifs

Comprendre les limites du modèle relationnel
Comprendre pourquoi et comment le modèle relationnel peut être étendu

a) Les atouts du modèle relationnel

- Fondé sur une théorie rigoureuse et des principes simples
- Mature, fiable, performant
- Indépendance programme et données
- SGBDR★ : les plus utilisés, connus, maîtrisés
- SQL★ une implémentation standard du modèle relationnel, avec des API★ pour la plupart des langages de programmation
- Les SGBDR incluent des outils performants de gestion de requêtes, de générateurs d'applications, d'administration, d'optimisation, etc.

b) Les inconvénients du modèle relationnel

- La structure de donnée en tables est pauvre d'un point de vue de la modélisation logique
- Le *mapping* MCD★ vers MLD★ entraîne une perte de sémantique
- La manipulation de structures relationnelles par des langages objets entraîne une *impedance mismatch*, c'est à dire un décalage entre les structures de données pour le

stockage et les structures de données pour le traitement (ce qui implique des conversions constantes d'un format à l'autre)

- La 1NF★ est inappropriée à la modélisation d'objets complexes
- La normalisation entraîne la genèse de structures de données complexes et très fragmentées, qui peuvent notamment poser des problèmes de performance ou d'évolutivité
- Le SQL doit toujours être combiné à d'autres langages de programmation pour être effectivement mis en œuvre
- La notion de méthode ne peut être intégrée au modèle logique, elle doit être gérée au niveau de l'implémentation physique
- Les types de données disponibles sont limités et non extensibles

c) Les SGBDOO

Introduction

Les SGBDOO★ ont été créés pour gérer des structures de données complexes, en profitant de la puissance de modélisation des modèles objets et de la puissance de stockage des BD★ classiques.

Objectifs des SGBDOO :

- Offrir aux langages de programmation orientés objets des modalités de stockage permanent et de partage entre plusieurs utilisateurs
- Offrir aux BD des types de données complexes et extensibles
- Permettre la représentation de structures complexes et/ou à taille variable

Avantages des SGBDOO :

- Le schéma d'une BD objet est plus facile à appréhender que celui d'une BD relationnelle (il contient plus de sémantique, il est plus proche des entités réelles)
- L'héritage permet de mieux structurer le schéma et de factoriser certains éléments de modélisation
- La création de ses propres types et l'intégration de méthodes permettent une représentation plus directe du domaine
- L'identification des objets permet de supprimer les clés artificielles souvent introduites pour atteindre la 3NF★ et donc de simplifier le schéma
- Les principes d'encapsulation et d'abstraction du modèle objet permettent de mieux séparer les BD de leurs applications (notion d'interface).

Inconvénient des SGBDOO :

- Gestion de la persistance et de la coexistence des objets en mémoire (pour leur manipulation applicative) et sur disque (pour leur persistance) complexe
- Gestion de la concurrence (transactions) plus difficile à mettre en œuvre
- Interdépendance forte des objets entre eux
- Gestion des pannes
- Complexité des systèmes (problème de fiabilité)
- Problème de compatibilité avec les SGBDR classiques

Fondamental

Les SGBDOO apportent des innovations sur des aspects que les SGBDR ne savent pas faire, mais sans être au même niveau sur ce que les SGBDR savent bien faire.

d) Les SGBDRO

Introduction

Les SGBDRO★ sont nés du double constat de la puissance nouvelle promise par les SGBDRO★ et de l'insuffisance de leur réalité pour répondre aux exigences de l'industrie des BD★ classiques. Leur approche est plutôt d'introduire dans les SGBDR★ les concepts apportés par les SGBDRO★ plutôt que de concevoir de nouveaux systèmes.

Objectifs des SGBDRO

- Gérer des données complexes (temps, géo-référencement, multimédia, types utilisateurs, etc.)
- Rapprocher le modèle logique du modèle conceptuel
- Réduire l'impédance mismatch☹
- Réduire les pertes de performance liées à la normalisation et aux jointures

Définition : Modèle relationnel-objet

Modèle relationnel étendu avec des principes objet pour en augmenter les potentialités.

Synonymes : Modèle objet-relationnel

Fondamental : Type utilisateur

Le concept central du RO est celui de **type utilisateur**, correspondant à la classe en POO★, qui permet d'injecter la notion d'objet dans le modèle relationnel.

Les apports principaux des types utilisateurs sont :

- **La gestion de l'imbrication (données structurées et collections)**
- **Les méthodes et l'encapsulation**
- **L'héritage et la réutilisation de définition de types**
- **L'identité d'objet et les références physiques**

Attention : Tables imbriquées et tables objets

Le modèle RO apporte deux nouveautés distinctes et indépendantes à ne pas confondre :

- **Les tables imbriquées (*nested model*) qui permettent de dépasser les limites de la première forme normale et de limiter la fragmentation de l'information ;**
- **Les tables objets qui permettent d'ajouter les identifiants d'objets (OID), les méthodes et l'héritage aux tables classiques.**

Avec un SGBDRO on peut ne pas utiliser ces deux extensions (on reste alors en relationnel), utiliser l'une des deux ou bien les deux conjointement.

e) Exercice

Parmi les assertions suivantes, lesquelles expriment des limites des SGBDR qui justifient l'évolution du modèle relationnel classique vers le modèle relationnel-objet ?

- Le principe d'atomicité des attributs d'une relation (première forme normale) empêche de disposer de propriétés structurant plusieurs valeurs dans un type complexe.
- La séparation des données et des traitements empêche l'intégration des méthodes au modèle.
- La représentation de données complexes conduit à une fragmentation importante de l'information en de multiples relations et à des chutes de performance.
- Le modèle relationnel ne permet pas d'exécuter correctement des transactions en réseau.
- Il n'est pas possible de créer des types de données personnalisées.
- Il n'est pas possible d'exécuter des instructions SQL1 ou SQL2 à partir d'un langage objet tel que Java, alors que c'est possible avec SQL3.

2. Extension du relationnel et du SQL

a) Extension du modèle logique relationnel : les types

Définition : Type

Type de données créé par le concepteur d'un schéma relationnel-objet, qui encapsule des données et des opérations sur ces données. Ce concept est à rapprocher de celui de classe d'objets.

Synonymes : Type de données utilisateur, Type de données abstrait

Syntaxe : Niveau logique

```

1 Type nom_type : <
2   attribut1:typeAttribut1,
3   attribut2:typeAttribut2,
4   ...
5   attributN:typeAttributN,
6 >
```

Complément

Les types des attributs d'un type peuvent également être des types utilisateurs.

Complément : Niveau logique avec méthodes

```

1 Type nom_type : <
2   attribut1:typeAttribut1,
3   attribut2:typeAttribut2,
4   ...
5   attributN:typeAttributN,
6   =methode1:(paramètres) typeRetourné1,
7   =methode2:(paramètres) typeRetourné2,
8   =...
9   =methodeN:(paramètres) typeRetournéN
10 >
```

b) Création de type en SQL3 sous Oracle (extension au LDD)

Syntaxe : Déclaration de type

```

1 CREATE TYPE nom_type AS OBJECT (
2   nom_attribut1 type_attribut1,
3   ...
4 );
5 /
6

```

Exemple : Création de tables d'objets (enregistrements avec OID)

```

1 CREATE TABLE t OF nom_type (
2   ...
3 )

```

Exemple : Usage des types dans les tables (modèle imbriqué)

```

1 CREATE TABLE t (
2   ...
3   nom_attribut nom_type,
4   ...
5 )

```

*Complément**Héritage et réutilisation de types**Méthodes de table d'objets*

c) Exemple : Cours et intervenants

Exemple : Modèle logique

pknom	prenom	bureau			liste-telephones	listes-specialites	
		centre	batiment	numero		Domaine	Technologie
Crozat	Stéphane	PG	K	256	0687990000	BD	SGBDR
					0912345678	Doc	XML
					0344231234	BD	SGBDRO
Vincent	Antoine	R	C	123	0344231235	IC	Ontologie
					0687990001	Base de données	SGBDRO

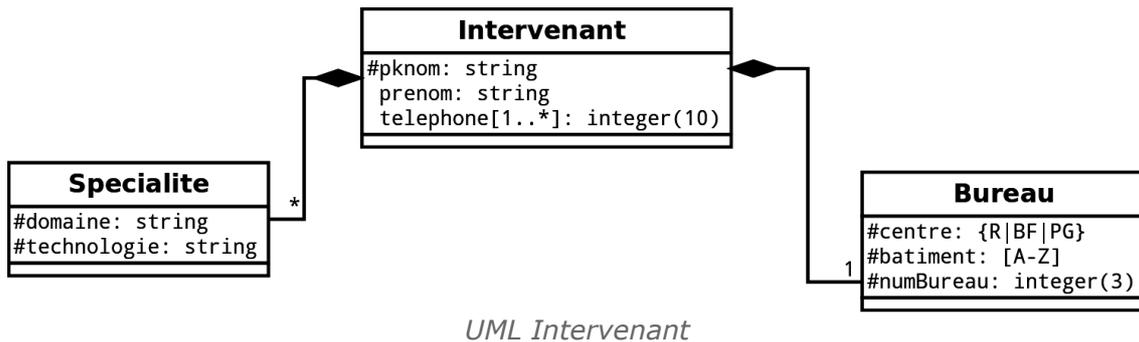
Modèle imbriqué

```

1 Type typBureau : <centre:char, batiment:char, numero:int>
2 Type typListeTelephones : collection de <entier>
3 Type typSpecialite : <domaine:char, specialite:char>
4 Type typListeSpecialites : collection de <typSpecialite>
5 tIntervenant (#nom:char, prenom:char, bureau:typBureau,
   ltelephones:typListeTelephones, lspecialites:typListeSpecialites)

```

Complément : Modèle conceptuel



Exemple : LDD

```

1 CREATE OR REPLACE TYPE typBureau AS OBJECT (
2   centre char(2),
3   batiment char(1),
4   numero number(3)
5 );
6 /
7
8 CREATE OR REPLACE TYPE typListeTelephones AS TABLE OF number(10);
9 /
10
11 CREATE OR REPLACE TYPE typSpecialite AS OBJECT (
12   domaine varchar2(15),
13   technologie varchar2(15)
14 );
15 /
16
17 CREATE OR REPLACE TYPE typListeSpecialites AS TABLE OF typSpecialite;
18 /
19
20 CREATE TABLE tIntervenant (
21   pknom varchar2(20) PRIMARY KEY,
22   prenom varchar2(20) NOT NULL,
23   bureau typBureau,
24   ltelephones typListeTelephones,
25   lspecialites typListeSpecialites
26 )
27 NESTED TABLE ltelephones STORE AS tIntervenant_nt1,
28 NESTED TABLE lspecialites STORE AS tIntervenant_nt2;
  
```

Exemple : Insert

```

1 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
2 VALUES (
3   'Crozat',
4   'Stéphane',
5   typBureau('PG', 'K', 256),
6   typListeTelephones (0687990000, 0912345678, 0344231234),
7   typListeSpecialites (typSpecialite ('BD', 'SGBDR'), typSpecialite ('Doc', 'XML'),
8     typSpecialite ('BD', 'SGBDRO'))
9 );
10 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
11 VALUES (
12   'Vincent',
13   'Antoine',
14   typBureau('R', 'C', 123),
15   typListeTelephones (0344231235, 0687990001),
  
```

```

16 typListeSpecialites (typSpecialite ('IC','Ontologies'),
17 typSpecialite('BD','SGBDRO'))
);

```

Exemple : Select (enregistrement imbriqué)

```
1 SELECT pknom, prenom, i.bureau.centre FROM tIntervenant i;
```

1	PKNOM	PRENOM	BUREAU.CENTRE
2	-----	-----	-----
3	Crozat	Stéphane	PG
4	Vincent	Antoine	R

Exemple : Select (collection de scalaires imbriquée)

```

1 SELECT i.pknom, t.*
2 FROM tIntervenant i, TABLE(i.ltelephones) t
3

```

1	PKNOM	COLUMN_VALUE
2	-----	-----
3	Crozat	687990000
4	Crozat	912345678
5	Crozat	344231234
6	Vincent	344231235
7	Vincent	687990001

Exemple : Select (collection d'enregistrements imbriquée)

```

1 SELECT i.pknom, s.*
2 FROM tIntervenant i, TABLE(i.lspecialites) s

```

1	PKNOM	DOMAINE	TECHNOLOGIE
2	-----	-----	-----
3	Crozat	BD	SGBDR
4	Crozat	Doc	XML
5	Crozat	BD	SGBDRO
6	Vincent	IC	Ontologies
7	Vincent	BD	SGBDRO

3. Les tables imbriquées (nested model)

a) Le modèle imbriqué

Définition : Modèle imbriqué

Le principe du modèle imbriqué est qu'un attribut d'une table ne sera plus seulement valué par une unique valeur scalaire (principe de la 1NF★), mais pourra l'être par un vecteur (enregistrement) ou une collection de scalaires ou de vecteurs, c'est à dire une autre table.

Synonyme : *nested model*

Fondamental : Finalité du modèle imbriqué

La 1NF★ est relâchée pour permettre d'affecter des valeurs non atomiques à un attribut afin de modéliser des objets complexes.

Définition : Table imbriquée

Table relationnel-objet dont chaque attribut peut être défini pour contenir :

- une variable scalaire,
- ou un enregistrement,
- ou une collection de scalaires,
- ou une collection enregistrements.

Exemple : Gestion d'attribut composé par imbrication d'enregistrement (table à une seule ligne)

Imbrication d'enregistrement

Exemple : Gestion d'attribut multivalué par imbrication de collection de scalaires (table à une seule colonne)

Imbrication de collection de scalaires

Exemple : Gestion de composition par imbrication de collection d'enregistrements (tables)

Imbrication de collection d'enregistrement

Exemple : Synthèse

Modèle imbriqué

b) Les enregistrements imbriqués (modèle logique)

Exemple : Gestion d'attribut composé par imbrication d'enregistrement (table à une seule ligne)

pknom	prenom	Bureau		
Crozat	Stéphane	centre	batiment	numero
		PG	K	256
Vincent	Antoine	centre	batiment	numero
		R	C	123

Imbrication d'enregistrement

Exemple : Modèle logique

```

1 Type typBureau : <centre:char, batiment:char, numero:int>
2 tIntervenant (#nom:char, prenom:char, bureau:typBureau)

```

Remarque : Première forme normale

Le recours aux types utilisateurs brise la règle de la première forme normale de l'atomicité des valeurs d'attribut.

Ce non respect de la 1NF★ ne pose pas de problème en relationnel-objet car la déclaration de type permet de contrôler la structure interne des enregistrements imbriqués.

Dans le modèle relationnel classique, le problème du non respect de la 1NF★ était bien l'opacité de la structure interne des attributs ainsi constitués qui interdisait l'accès à des sous informations extraites de la valeur de l'attribut (par exemple un attribut comprenant le nom et le prénom ensemble interdit l'accès à l'information "nom" ou à l'information "prénom" indépendamment). L'usage de type éclaire cette opacité et rend ainsi le respect de l'atomicité dépassable.

c) Les collections imbriquées (modèle logique)

Définition : Collection

Une collection est un type de données générique défini afin de supporter un ensemble d'objets (scalaires ou enregistrements).

Synonymes : Collection d'objets

Syntaxe : Modèle logique

```

1 Type nom_type : collection de <type_objet>

```

Remarque

Les objets d'une collection peuvent être un type simple (collection de scalaire) ou un type utilisateur (collection d'enregistrements).

Exemple : Collection d'entiers (modèle logique)

```

1 Type typListeTelephones : collection de <entier>
2 tIntervenant (#nom:char, prenom:char, ltelephones:typListeTelephones)

```

Exemple : Collection d'objets complexes (modèle logique)

```

1 Type typSpecialite : <domaine:char, specialite:char>
2 Type typListeSpecialites : collection de <typSpecialite>
3 tIntervenant (#nom:char, prenom:char, lspecialites:typListeSpecialites)

```

4. Apport du modèle imbriqué au passage conceptuel-logique

Objectifs

Savoir déduire un modèle logique relationnel-objet mobilisant l'imbrication depuis un modèle conceptuel UML.

Intégrer les apports du modèle relationnel-objet par rapport au relationnel dans ce processus.

Le modèle imbriqué permet de traiter de façon plus naturelle les attributs composites et multivalués et les compositions.

Dans ces deux cas, on remplace la méthode utilisée pour le relationnel.

a) Attributs composés et multivalués

Attributs composés

Pour chaque type d'attribut composé, créer un type d'objet.

Attributs multi-valués

Pour chaque attribut multivalué créer une collection du type de cet attribut.

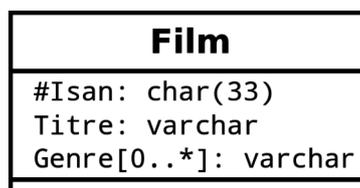
Remarque : Attributs composés multivalués

Si l'attribut multivalué est composite, alors créer une collection d'objets.

b) Un film avec des attributs

Question

Proposer un modèle RO correspondant au modèle UML.



Film (attribut multi-valué)

c) Composition

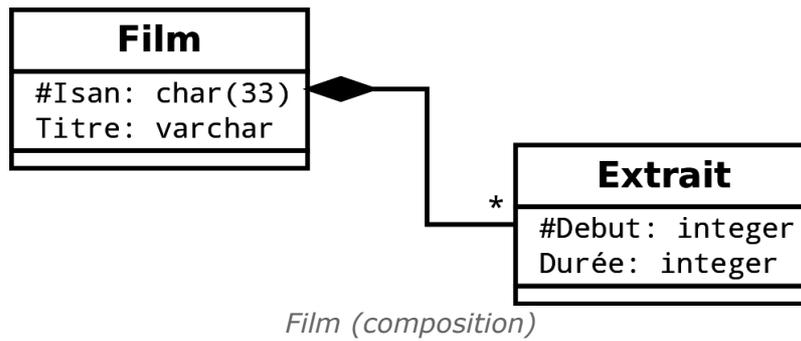
En RO, les compositions sont gérées avec le modèle imbriqué, à l'instar des attributs composés et multi-valués : en effet par définition l'élément composant ne peut être partagé par plusieurs composites, le modèle imbriqué n'introduit donc pas de redondance.

1. Créer un type correspondant au schéma de la table des composants
2. Créer une collection de ce type
3. Imbriquer cette collection dans la table des composites

d) Un film bien composé

Question

Proposer un modèle RO correspondant au modèle UML.

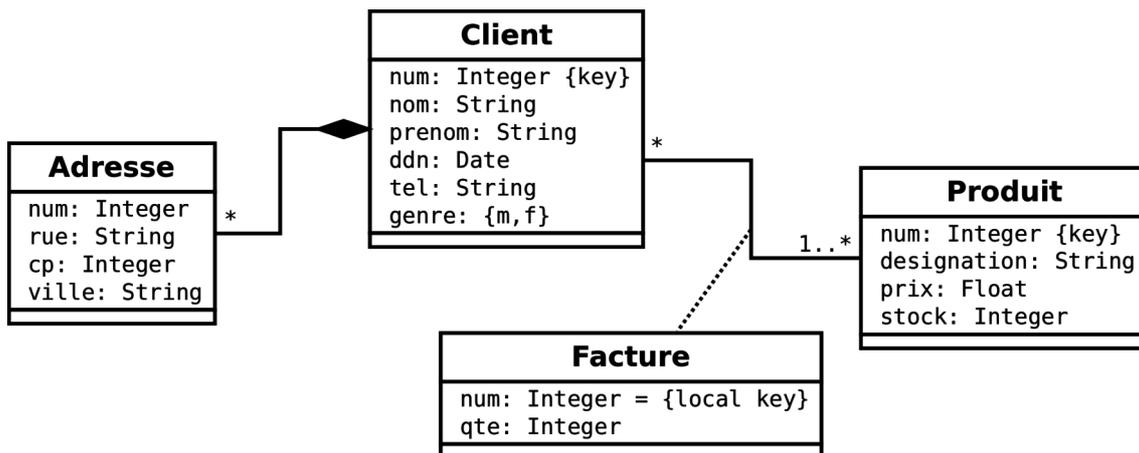


B. Exercices

1. MediaTek I

[15 min]

L'association **MediaTek** souhaite réaliser la base de données correspondant au schéma UML ci-après en utilisant un modèle relationnel-objet ne mobilisant que les tables imbriquées (*nested model*).

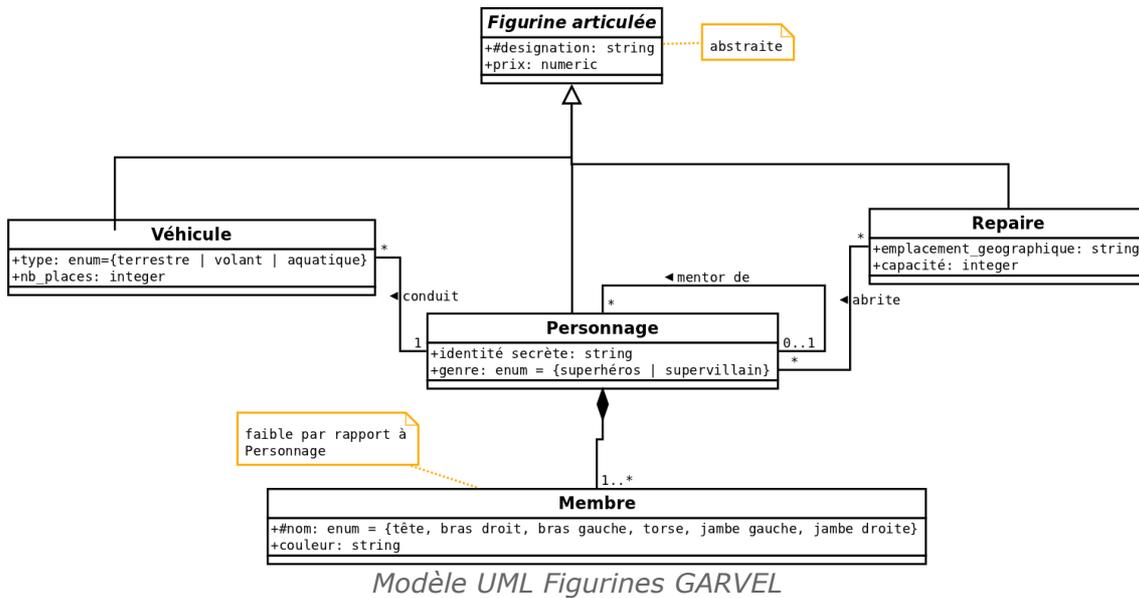


Question

Réaliser le passage vers un modèle RO, en utilisant le modèle imbriqué.

2. Super-héros relationnels-objets imbriqués

[15 minutes]



Question

Transformer le modèle UML en modèle relationnel-objet. On utilisera uniquement le modèle imbriqué.

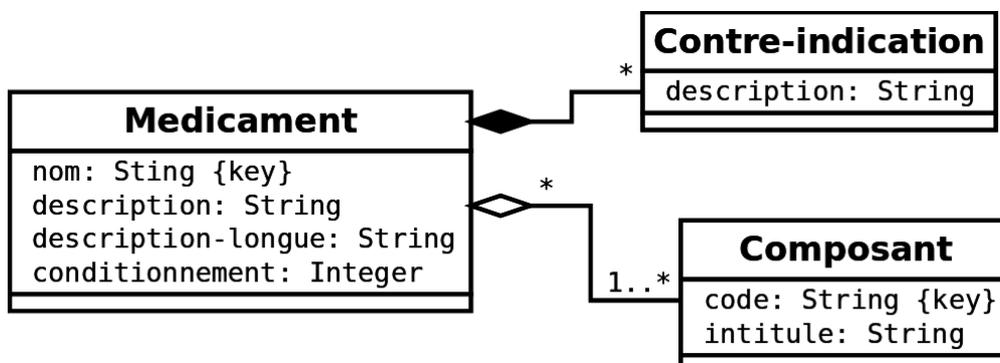
C. Devoirs

1. Lab V

15 min

Réaliser le passage RO de ce modèle UML en utilisant le modèle imbriqué.

On proposera de considérer *description* et *description-longue* comme deux éléments d'un attribut composé (bien que ce ne soit pas représenté sur le schéma UML).

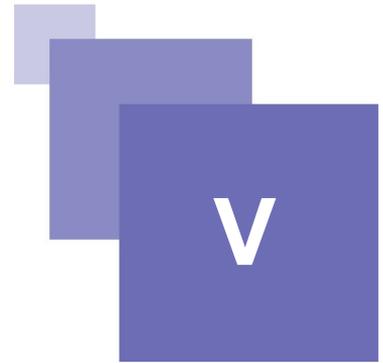


2. Arbre de scène 3D II

45 min

1	# Scènes et objets
2	- Les scènes sont identifiées de manière unique par un nom.
3	- Une scène peut être active ou inactive.
4	- Une scène contient des objets qui eux-mêmes peuvent appartenir à plusieurs scènes, au moins une.
5	- Dans chaque scène, les objets peuvent être visibles ou invisibles.
6	- On souhaite pouvoir calculer le nombre de faces affichées pour une scène donnée (Méthode).
7	
8	# Objets
9	- Les objets sont identifiés de manière unique par un nom.
10	- Un objet a comme propriété une position dans l'espace, représentée par un vecteur de réels à trois
11	composantes x , y, z (attribut composé ou dataType) .
12	- Les objets sont organisés de manière hiérarchique : un objet peut être parent de plusieurs objets et chaque objet peut avoir au plus un parent (association réflexive).
13	- Un personnage est un objet particulier qui possède des animations. Une animation est identifiée de manière locale par un nom (composition).
14	
15	# Scripts
16	- Des scripts peuvent être associés à un objet ou à une scène, à l'un ou à l'autre mais pas aux deux (XOR).
17	- Un script est identifié de manière unique par un nom et possède également un attribut permettant de connaître son état actif ou inactif.
18	
19	# Maillage et éléments
20	- À un objet est associé un maillage et celui-ci peut être utilisé par plusieurs objets.
21	- Un maillage est identifié de manière unique par un nom et est composé de plusieurs éléments. Chaque élément est identifié de manière locale par un numéro (Composition).
22	- Il existe exactement trois types d'élément : des nœuds, des arrêtes et des faces (Héritage).
23	- Une face est délimitée par trois arrêtes et chaque arrête est délimitée par deux nœuds. Plusieurs arrêtes peuvent partager un même nœud et plusieurs faces peuvent partager une même arrête.
24	
25	# Matériau et texture
26	- Un maillage possède plusieurs matériaux identifiés de manière unique par un nom. Un matériau peut être associé à plusieurs maillages.
27	- Un matériau est caractérisé par une couleur définie par un vecteur d'entiers à quatre composantes : rouge, vert, bleu, alpha.
28	- Un matériau peut posséder une texture et celle-ci peut être utilisée par plusieurs matériaux. Une texture est identifiée de manière unique par un nom et possède comme attribut une image.

Tables imbriquées en relationnel-objet sous Oracle



A. Cours

1. Imbrication en SQL3

Objectifs

Connaître l'extension du LDD pour déclarer des tables imbriquées
Connaître l'extension du LMD pour naviguer des tables imbriquées

a) Les enregistrements imbriqués (LDD)

Rappel

Les enregistrements imbriqués (modèle logique)

Rappel

Création de type en SQL3 sous Oracle (extension au LDD)

Exemple : Implémentation SQL3

```
1 CREATE OR REPLACE TYPE typBureau AS OBJECT (  
2   centre char(2),  
3   batiment char(1),  
4   numero number(3)  
5 );  
6 /  
7  
8 CREATE TABLE tIntervenant (  
9   pknom varchar2(20) PRIMARY KEY,  
10  prenom varchar2(20) NOT NULL,  
11  bureau typBureau  
12 );
```

b) Insertion dans les enregistrements imbriqués

En RO, la manipulation de données est étendue pour assurer la gestion des objets et des collections.

Fondamental : Constructeur d'objet

Pour tout type d'objet est automatiquement créée une méthode de construction, dont le nom est celui du type et les arguments les attributs du type.

Syntaxe : Insertion d'enregistrements imbriqués

```
1 INSERT INTO nom_table (... , attribut_type_objet, ...)
2 VALUES
3 (... , nom_type(valeur1, valeur2, ...), ...);
```

Exemple : Insertion d'enregistrements imbriqués

```
1 INSERT INTO tIntervenant (pknom, prenom, bureau)
2 VALUES ('Crozat', 'Stéphane', typBureau('PG','K',256));
3
4 INSERT INTO tIntervenant (pknom, prenom, bureau)
5 VALUES ('Vincent', 'Antoine', typBureau('R','C',123));
6
```

c) Sélection dans les enregistrements imbriqués

Syntaxe : Accès aux attributs des enregistrements imbriqués

```
1 SELECT alias_table.nom_objet.nom_attribut
2 FROM nom_table AS alias_table
```

Attention : Alias obligatoire

L'accès aux objets se fait obligatoirement en préfixant le chemin d'accès par l'alias de la table et non directement par le nom de la table.

Exemple : Accès aux attributs des enregistrements imbriqués

```
1 SELECT pknom, prenom, i.bureau.centre FROM tIntervenant i;
```

1	PKNOM	PRENOM	BUREAU.CENTRE
2	-----	-----	-----
3	Crozat	Stéphane	PG
4	Vincent	Antoine	R

Complément : Accès aux méthodes des objets

```
1 SELECT alias_table.nom_objet.nom_méthode(paramètres)
2 FROM nom_table AS alias_table
```

d) Les collections imbriquées (LDD)

Rappel

Les collections imbriquées (modèle logique)

Syntaxe : Implémentation des collections de scalaires sous forme de tables imbriquées en SQL3

```

1  -- Déclaration d'un type abstrait de collection
2  CREATE TYPE liste_nom_type AS TABLE OF nom_type_scalaire;
3
4  -- Déclaration d'une table imbriquant une autre table
5  CREATE TABLE nom_table_principale (
6    nom_attribut type_attribut,
7    ...
8    nom_attribut_table_imbriquée liste_nom_type,
9    ...
10 )
11 NESTED TABLE nom_attribut_table_imbriquée STORE AS nom_table_stockage;

```

Exemple : Gestion d'attribut multivalué par imbrication de collection de scalaires

pknom	prenom	liste-telephones
Crozat	Stéphane	0687990000 0912345678 0344231234
Vincent	Antoine	0344231235 0687990001

Imbrication de collection de scalaires

```

1  CREATE OR REPLACE TYPE typListeTelephones AS TABLE OF number(10);
2  /
3
4  CREATE TABLE tIntervenant (
5    pknom varchar2(20) PRIMARY KEY,
6    prenom varchar2(20) NOT NULL,
7    ltelephones typListeTelephones
8  )
9  NESTED TABLE ltelephones STORE AS tIntervenant_ntl;

```

Syntaxe : Implémentation des collections d'enregistrement sous forme de tables imbriquées en SQL3

```

1  --Création d'un type abstrait d'objet
2  CREATE TYPE nom_type AS OBJECT (...);
3
4  -- Déclaration d'un type abstrait de collection
5  CREATE TYPE liste_nom_type AS TABLE OF nom_type;
6
7  -- Déclaration d'une table imbriquant une autre table
8  CREATE TABLE nom_table_principale (
9    nom_attribut type_attribut,

```

```

10 ...
11 nom_attribut_table_imbriquée liste_nom_type,
12 ...
13 )
14 NESTED TABLE nom_attribut_table_imbriquée STORE AS nom_table_stockage;

```

Exemple : Gestion de composition par imbrication de collection d'enregistrements

pknom	pre nom	liste-specialites	
Crozat	Stéphane	Domaine	Technologie
		BD	SGBDR
		Doc	XML
		BD	SGBDRO
Vincent	Antoine	Domaine	Technologie
		IC	Ontologie
		Base de données	SGBDRO

Imbrication de collection d'enregistrement

```

1 CREATE OR REPLACE TYPE typSpecialite AS OBJECT (
2   domaine varchar2(50),
3   technologie varchar2(50)
4 );
5 /
6
7 CREATE OR REPLACE TYPE typListeSpecialites AS TABLE OF typSpecialite;
8 /
9
10 CREATE TABLE tIntervenant (
11   pknom varchar2(20) PRIMARY KEY,
12   pre nom varchar2(20) NOT NULL,
13   lspecialites typListeSpecialites
14 )
15 NESTED TABLE lspecialites STORE AS tIntervenant_nt2;

```

Complément : Synthèse : enregistrement imbriqué, collection de scalaires imbriquée, collection d'enregistrement imbriquée

Modèle imbriqué

```

1 Type typBureau : <centre:char, batiment:char, numero:int>
2 Type typListeTelephones : collection de <entier>
3 Type typSpecialite : <domaine:char, specialite:char>
4 Type typListeSpecialites : collection de <typSpecialite>
5 tIntervenant (#nom:char, prenom:char, bureau:typBureau,
   ltelephones:typListeTelephones, lspecialites:typListeSpecialites)

```

```

1 CREATE OR REPLACE TYPE typBureau AS OBJECT (
2   centre char(2),
3   batiment char(1),
4   numero number(3)
5 );
6 /
7
8 CREATE OR REPLACE TYPE typListeTelephones AS TABLE OF number(10);
9 /
10
11 CREATE OR REPLACE TYPE typSpecialite AS OBJECT (
12   domaine varchar2(15),
13   technologie varchar2(15)
14 );
15 /
16
17 CREATE OR REPLACE TYPE typListeSpecialites AS TABLE OF typSpecialite;
18 /
19
20 CREATE TABLE tIntervenant (
21   pknom varchar2(20) PRIMARY KEY,
22   prenom varchar2(20) NOT NULL,
23   bureau typBureau,
24   ltelephones typListeTelephones,
25   lspecialites typListeSpecialites
26 )
27 NESTED TABLE ltelephones STORE AS tIntervenant_nt1,
28 NESTED TABLE lspecialites STORE AS tIntervenant_nt2;

```

e) Insertion dans les collections imbriquées

Syntaxe : Insertion de collections d'objets

```

1 INSERT INTO nom_table (... , attribut_type_collection, ...)
2 VALUES (
3   ...
4   nom_type_collection(
5     nom_type_objet(valeur1, valeur2, ...),
6     nom_type_objet(valeur1, valeur2, ...),
7     ...);
8   ...
9 );

```

Exemple : Insertion d'enregistrements et de collections imbriqués

```

1 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
2 VALUES (
3   'Crozat',
4   'Stéphane',
5   typBureau('PG', 'K', 256),
6   typListeTelephones (0687990000, 0912345678, 0344231234),
7   typListeSpecialites (typSpecialite ('BD', 'SGBDR'), typSpecialite ('Doc', 'XML'),
8   typSpecialite ('BD', 'SGBDRO'))
9 );
10 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)

```

```

11 VALUES (
12 'Vincent',
13 'Antoine',
14 typBureau('R','C',123),
15 typListeTelephones (0344231235,0687990001),
16 typListeSpecialites (typSpecialite ('IC','Ontologies'),
17 typSpecialite('BD','SGBDRO'))
);

```

Complément

Instruction THE

f) Sélection dans les collections imbriquées

Syntaxe : Accès aux tables imbriquées

Soit `col` un attribut de la table `t` contenant une collection imbriquée.

```

1 SELECT t2.*
2 FROM t t1, TABLE(t1.col) t2;

```

Attention : Jointure avec la table imbriquée

La jointure entre la table principale et sa table imbriquée est implicitement réalisée, il ne faut pas la spécifier.

Exemple : Accès à une collection imbriquée de scalaires

```

1 SELECT i.pknom, t.*
2 FROM tIntervenant i, TABLE(i.ltelephones) t
3

```

1	PKNOM	COLUMN_VALUE
2	-----	-----
3	Crozat	687990000
4	Crozat	912345678
5	Crozat	344231234
6	Vincent	344231235
7	Vincent	687990001

Exemple : Accès à une collection imbriquée d'enregistrement

```

1 SELECT i.pknom, s.*
2 FROM tIntervenant i, TABLE(i.lspecialites) s

```

1	PKNOM	DOMAINE	TECHNOLOGIE
2	-----	-----	-----
3	Crozat	BD	SGBDR
4	Crozat	Doc	XML
5	Crozat	BD	SGBDRO
6	Vincent	IC	Ontologies
7	Vincent	BD	SGBDRO

Syntaxe : Accès aux colonnes d'une table imbriquée

```

1 SELECT t2.a, t2.b...
2 FROM t t1, TABLE(t1.col) t2;

```

Syntaxe : Accès à la colonne d'une table imbriquée de scalaires

Lorsque la table imbriquée est une table de type scalaire (et non une table d'objets d'un type utilisateur), alors la colonne de cette table n'a pas de nom (puisque le type est scalaire la table n'a qu'une colonne). Pour accéder à cette colonne, il faut utiliser une syntaxe dédiée : COLUMN_VALUE.

```
1 SELECT t2.COLUMN_VALUE
2 FROM t t1, TABLE(t1.nt) t2;
```

Exemple

```
1 SELECT i.pknom, t.COLUMN_VALUE, s.domaine
2 FROM tIntervenant i, TABLE(i.ltelephones) t, TABLE(i.lspecialites) s
```

1	PKNOM	COLUMN_VALUE	DOMAINE
2	-----	-----	-----
3	Crozat	687990000	BD
4	Crozat	687990000	Doc
5	Crozat	687990000	BD
6	Crozat	912345678	BD
7	Crozat	912345678	Doc
8	Crozat	912345678	BD
9	Crozat	344231234	BD
10	Crozat	344231234	Doc
11	Crozat	344231234	BD
12	Vincent	344231235	IC
13	Vincent	344231235	BD
14	Vincent	687990001	IC
15	Vincent	687990001	BD

2. Compléments SQL3 pour le modèle imbriqué

a) Extension THE : Manipulation des collections imbriquées

Définition : THE

La clause THE est une extension du LMD★ permettant de manipuler les objets (scalaires ou enregistrements) dans les collections implémentées sous forme de tables imbriquées.

Syntaxe

- INSERT INTO THE (SELECT ...) VALUES (...)
- DELETE THE (SELECT ...) WHERE ...
- UPDATE THE (SELECT ...) SET ... WHERE ...

Exemple : INSERT d'un scalaire ou d'un enregistrement dans une collection

```
1 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
2 VALUES (
3 'Dumas',
4 'Leonard',
5 typBureau('R','C',123),
6 typListeTelephones(0344234423),
7 typListeSpecialites()
8 );
9
10 INSERT INTO THE (SELECT i.ltelephones FROM tIntervenant i WHERE i.pknom='Dumas')
11 VALUES (0666666666);
12
```

```

13 INSERT INTO THE (SELECT i.lspecialites FROM tIntervenant i WHERE i.pknom='Dumas')
14 VALUES (typSpecialite('BD','SGBDR'));

```

Remarque

L'emploi de `typListeSpecialites()` permet de déclarer une collection imbriquée vide dans `tIntervenant`. La collection ne contient aucun élément initialement, ils peuvent être ajoutés ultérieurement grâce à l'instruction `INSERT INTO THE`.

Exemple : DELETE d'un objet dans une collection

```

1 DELETE THE (SELECT i.ltelephones FROM tIntervenant i WHERE i.pknom='Dumas') nt
2 WHERE nt.COLUMN_VALUE=0344234423;

```

Exemple : UPDATE d'un objet dans une collection

```

1 UPDATE THE (SELECT i.lspecialites FROM tIntervenant i WHERE i.pknom='Dumas') nt
2 SET nt.technologie='SGBDRO'
3 WHERE nt.domaine='BD';

```

b) Insertion de collections à partir de requêtes SELECT

Les constructeurs `⊃` d'objets sont utilisables dans les instructions de type `INSERT ... VALUES`, mais ne peuvent être utilisés dans les instructions de type `INSERT ... SELECT`. Pour ces dernières une autre méthode de construction est nécessaire.

Syntaxe : Insérer une collection d'objets depuis une table

Soient les tables `tro` et `t` définies par le schéma RO ci-après.

```

1 type col : collection de <string>
2 tro(a_obj:col)
3 t(a:string)

```

L'instruction suivante permet d'insérer le contenu de l'attribut `a` pour tous les enregistrements de `t` dans un seul attribut `a_obj` d'un seul enregistrement de `tro` sous la forme d'une collection.

```

1 INSERT INTO tro (a_objet)
2 VALUES (CAST(MULTISET(SELECT a FROM t ) AS col));

```

- L'instruction `CAST (...) AS <type>` permet de renvoyer un type donné à partir d'une expression.
- L'instruction `MULTISET (...)` permet de renvoyer une collection à partir d'une liste de valeurs.

Méthode : Approche générale

Soit la table `tro` définie par le schéma RO ci-après. Elle contient un attribut clé (`pk_id`) et un attribut qui est une collection (`a_obj`).

```

1 type col : collection de <string>
2 tro(#pk_id:string, a_obj:col)

```

Soit la table `tr` définie par le schéma relationnel ci-après.

```

1 tr(a:string, b:string)

```

a1	b1
----	----

a1	b2
a1	b3
a2	b4
a2	b5

Tableau 2 Extrait de tr

Si l'on souhaite insérer le contenu de tr dans tro de telle façon que les valeurs de a correspondent à pk_id et celles de b à obj_a, en insérant une ligne pour chaque valeur a_x et que tous les b_x correspondant soient stockés dans la même collection, il faut exécuter une requête INSERT ... SELECT qui fait appel aux instructions CAST et MULTISET.

```

1 INSERT INTO tro (pk_id, a_obj)
2 SELECT
3     a,
4     CAST(
5         MULTISET(
6             SELECT b
7             FROM tr tr1
8             WHERE tr1.a= tr2.a)
9         AS col)
10 FROM tr tr2;
```

a1	(b1, b2, b3)
a2	(b4, b5)

Tableau 3 Extrait de tro après exécution de la requête INSERT

c) THE

[45 min]

Soit la séquence d'instructions suivante utilisant la clause THE :

```

1 INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
2 VALUES (
3     'Dumas',
4     'Leonard',
5     typBureau('R','C',123),
6     typListeTelephones(0344234423),
7     typListeSpecialites()
8 );
9
10 INSERT INTO THE (SELECT i.ltelephones FROM tIntervenant i WHERE i.pknom='Dumas')
11 VALUES (0666666666);
12
13 INSERT INTO THE (SELECT i.lspecialites FROM tIntervenant i WHERE i.pknom='Dumas')
14 VALUES (typSpecialite('BD','SGBDR'));

1 DELETE THE (SELECT i.ltelephones FROM tIntervenant i WHERE i.pknom='Dumas') nt
2 WHERE nt.COLUMN_VALUE=0344234423;

1 UPDATE THE (SELECT i.lspecialites FROM tIntervenant i WHERE i.pknom='Dumas') nt
2 SET nt.technologie='SGBDRO'
3 WHERE nt.domaine='BD';
```

Question 1

Rétro-concevez le modèle RO permettant les instructions précédentes.

Produisez la syntaxe MLD★ et SQL★.

Indice :

Rappel : Extension THE

Question 2

Rétro-concevez le modèle UML associé.

Question 3

Que renvoie l'instruction suivante :

```
1 SELECT i.pknom, s.*, t.*
2 FROM tIntervenant i, TABLE(i.lspecialites) s, TABLE(i.ltelephones) t
3 WHERE i.pknom='Dumas'
```

Indice :

Vous dessinerez un tableau avec les entêtes de colonne adéquates.

Question 4

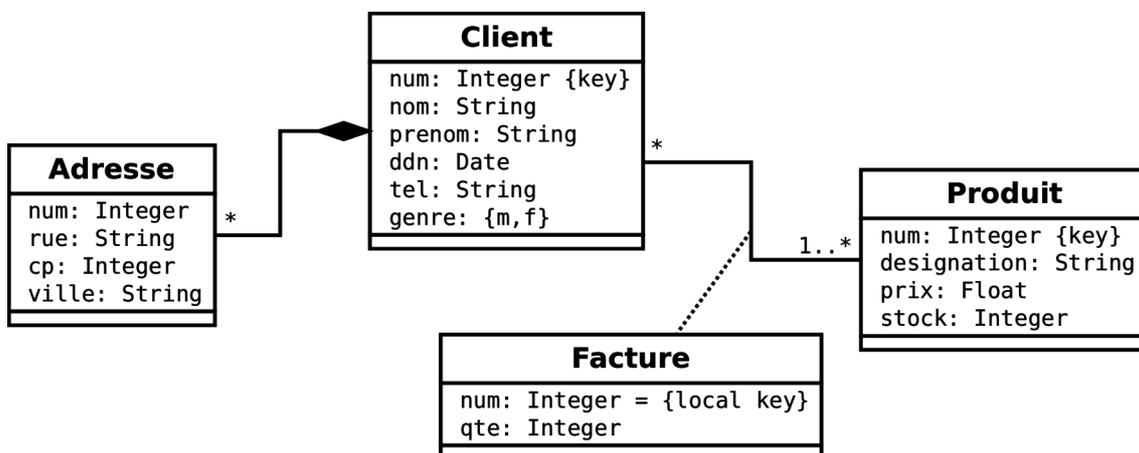
Changer le numéro de téléphone de Dumas avec la valeur de votre choix.

B. Exercices

1. MediaTek II

[45 min]

L'association **MediaTek** souhaite réaliser la base de données correspondant aux schémas ci-après.



```
1 Type Adresse : <num:integer, rue:string, cp:integer, ville:string>
2 Type ListeAdresse : collection de <Adresse>
3 Client (#num:integer, nom:string, prenom:string, adresse:ListeAdresse, ddn:date,
tel:string, genre: {'m', 'f'})
4 Produit (#num:integer, designation:string, prix:float, stock:integer)
5 Facture (#num:integer, #client=>Client(num), #produit=>Produit, qte:integer)
```

Question 1

Réalisez l'implémentation SQL3 sous Oracle.

Alimenter une base RO

Une fois la base créée, insérer des données pour l'alimenter.

Question 2

Initialiser les tables Client et Produit avec les données de votre choix (au moins deux clients et deux produits, au moins un client avec deux adresses).

Indice :

Insertion dans les enregistrements imbriqués

Insertion dans les collections imbriquées

Question 3

Initialiser la table facture avec les données de votre choix (deux factures de deux lignes chacune au moins).

Question 4

Affichez le contenu des trois tables de la base de données.

Indice :

Sélection dans les collections imbriquées

*SQL*Plus*

Interroger une base RO : Utilisation des tables imbriquées

Nous allons à présent interroger la base RO imbriquée.

Question 5

Écrivez une requête permettant de renvoyer le nombre de produits total achetés par chaque client, ainsi que la moyenne de produit par facture.

Indice :

Il faut utiliser un regroupement.

Question 6

Écrivez une requête permettant de renvoyer les numéros et noms des clients de la ville de Compiègne ayant payé au moins une facture avec au moins deux articles (vous pouvez utiliser une autre ville correspondant à vos données) .

Indice :

Faites un regroupement sur les numéros de client, noms des client et numéros de facture.

Question 7

Créer une vue ClientR qui affiche le contenu de la table Client en mode relationnel (en supprimant l'imbrication donc).

Indice :

Sélection dans les collections imbriquées

*SQL*Plus*

2. RO sans fil imbriqué

[30 minutes]

L'on souhaite réaliser une base de données permettant de gérer tous les relais Wifi sur un site d'entreprise. Chaque relais est caractérisé par une coordonnée géographique (de type CooT, dont le code SQL3 est fourni ci-après) et fait référence à un modèle. On associe également à chaque relais son prix d'achat. Chaque modèle possède un type qui l'identifie de façon unique, une marque et une puissance.

```

1 CREATE TYPE CooT AS OBJECT (
2 latitude DECIMAL(5,3),
3 longitude DECIMAL(5,3)
4 );

```

Question 1

Proposez une implémentation RO SQL3 sous Oracle exploitant le modèle imbriqué. Vous proposerez un MCD et un MLD préalablement pour vous aider.

Question 2

Insérer les données suivantes dans votre base de données :

- Un modèle de marque SuperWif et de type X1, puissance 25mW
- Deux relais de ce modèle respectivement aux coordonnées (48.853 ; 2.35) et (48.978 ; 3.01), achetés chacun 100€.

Question 3

Écrivez deux requêtes permettant de renvoyer respectivement :

- La puissance du relais situé à la coordonnée (48.853,2.35)
- La moyenne des prix des relais pour chaque modèle (type et marque)

C. Devoirs

1. Document sous licence Creative Commons

[1h]

On souhaite créer un langage permettant de d'assembler des documents situés dans une base de données. Un document est décrit par un titre, un auteur, un nombre de pages et une licence *creative commons* (CC) : cc-by, cc-by-nc, cc-by-nd, cc-by-sa, cc-by-nc-nd, cc-by-nc-sa.

Un document peut être :

- soit un document maître, il dispose alors toujours d'un numéro d'enregistrement unique dans la base, et son titre sera toujours renseigné ;
- soit un document esclave, il est alors membre d'un document maître.

Question 1

Proposez en UML un type de données utilisateur, en utilisant le stéréotype `<<dataType>>`, afin de représenter une licence CC en utilisant trois booléens (sachant que toutes les licences intègrent nécessairement le *by*). Une licence cc-by aura donc trois valeurs 0.

	nc	nd	sa
cc-by	0	0	0

	nc	nd	sa
cc-by-nc	1	0	0
cc-by-nd	0	1	0
cc-by-sa	0	0	1
cc-by-nc-nd	1	1	0
cc-by-nc-sa	1	0	1

Tableau 4 Les 6 licences CC représentées par 3 booléens

Proposez une modélisation UML complète du problème exploitant ce type de données.

Indice :

<http://creativecommons.fr/licences¹⁷>

Question 2

Proposez une représentation RO avec une seule table, en utilisant le modèle imbriqué (vous proposerez un modèle logique ou bien un code SQL compatible Oracle).

Insérer les données suivantes dans votre table :

- Le document maître "Introduction à la blockchain" numéro 1, de 10 pages avec une licence cc-by.
- Deux documents esclaves de 5 pages chacun, avec une licence cc-by-sa.

Question 3

Proposez une **vue** affichant les documents maîtres qui n'ont aucune de licence nc (ni à leur niveau, ni au niveau de leurs esclaves)

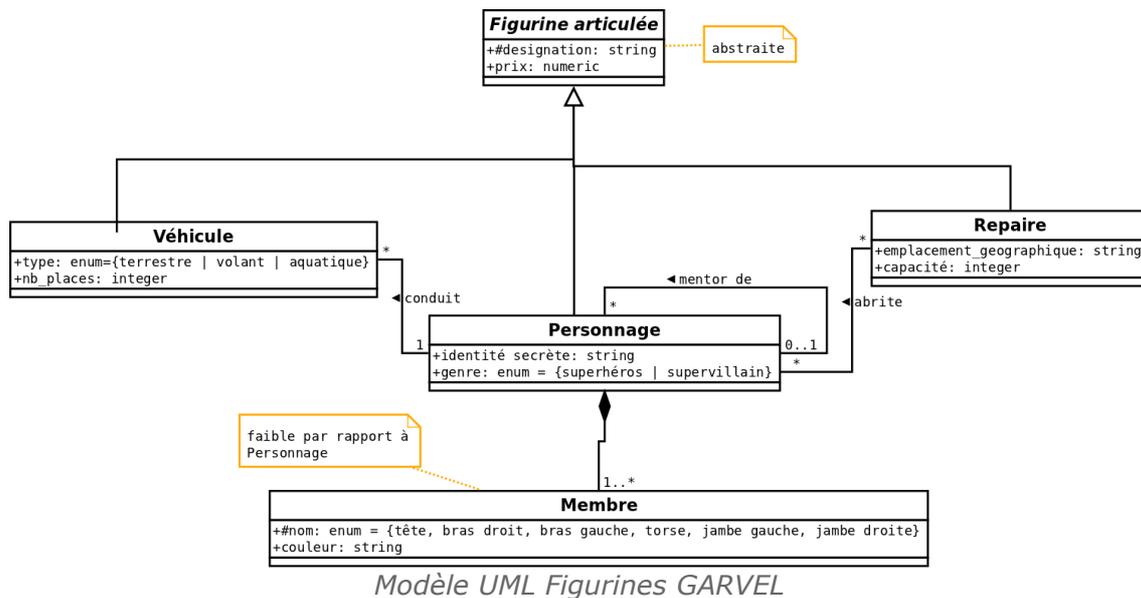
Question 4

Proposez une **vue** affichant les documents maîtres avec le nombre de pages total, sachant que ce nombre est égal à leur nombre de pages propre, plus le nombre de pages de tous leurs documents esclaves.

17 - <http://creativecommons.fr/licences>

2. Super-héros relationnels-objets imbriqués, épisode II

[20 minutes]



```

1  -- Personnage
2
3  Type typeMembre : <nom, couleur>
4  Type listeMembres : collection de typeMembre
5
6  Personnage (#designation, prix, identite_secrete, genre, mentor => Personnage,
7  membres:listeMembres)
8
9  -- Véhicule
10 Vehicule (#designation, prix, type, nb_places, personnage=>Personnage)
11
12 -- Repaire
13 Repaire (#designation, prix, emplacement_geographique, capacite)
14 Abrite (#repaire => Repaire, #personnage => Personnage)
  
```

Question 1

Donner, **en SQL**, la couleur des torsos des personnages habitant la "GARVEL Tower", pilotant des véhicules aquatiques et ayant comme mentor Superman.

Question 2

Implémentez le modèle RO en SQL3 sous Oracle, initialiser les données et tester la requête.

Tables d'objets en relationnel-objet

VI

A. Cours

Si le modèle logique relationnel a prouvé sa puissance et sa fiabilité depuis les années 1980, les nouveaux besoins de l'informatique industrielle ont vu l'émergence de structures de données complexes mal adaptées à une gestion relationnelle. La naissance du courant *orienté objet* et des langages associées (Java et C++ par exemple) ont donc également investi le champ des SGBD★ afin de proposer des solutions pour étendre les concepts du relationnel et ainsi mieux répondre aux nouveaux besoins de modélisation.

Nous étudions en particulier dans ce module les tables d'objets et les OID, c'est à dire la possibilité de doter chaque enregistrement d'un identifiant d'objet (OID) et de l'utiliser comme système de référencement à la place des clés étrangères (référence à un OID).

1. Tables d'objets et identifiants d'objets (OID)

a) Création de table d'objets (modèle logique)

Définition

Une table peut être définie en référençant un type de données plutôt que par des instructions LDD★ classiques. On parle alors de table objet.

Synonymes : table-objet, table d'objets

Syntaxe : Modèle logique

```
1 type nom_type : <...>
2 nom_table de nom_type (#attributs_clés) autres contraintes
```

Fondamental : OID

Les enregistrements d'une table-objet peuvent être identifiés par un OID★

Complément : Méthodes

Des méthodes peuvent être associées à une table-objet.

Complément : Héritage

Cette modalité de définition de schéma permet de profiter de l'héritage de type pour permettre l'héritage de schéma de table.

b) Les OID (identificateurs d'objets)

Le modèle relationnel-objet permet de disposer d'identificateurs d'objet (OID★).

Une table objet dispose d'un OID pour chacun de ses tuples.

Caractéristiques

- L'OID est une référence unique pour toute la base de données qui permet de référencer un enregistrement dans une table objet.
- L'OID est une référence physique (adresse disque) construit à partir du stockage physique de l'enregistrement dans la base de données.

Avantages

- Permet de créer des associations entre des objets sans effectuer de jointure (gain de performance).
- Fournit une identité à l'objet indépendamment de ses valeurs (clé artificielle).
- Fournit un index de performance maximale (un seul accès disque).
- Permet de garder en mémoire des identificateurs uniques dans le cadre d'une application objet, et ainsi gérer la persistance d'objets que l'on ne peut pas garder en mémoire, avec de bonnes performances (alors que sinon il faudrait exécuter des requêtes SQL pour retrouver l'objet).

Inconvénient

- Plus de séparation entre le niveau logique et physique.
- L'adresse physique peut changer si le schéma de la table change (changement de la taille d'un champ par exemple)
- Manipulation des données plus complexes, il faut un langage applicatif au dessus de SQL pour obtenir, stocker et gérer des OID dans des variables.

Méthode : Cadre d'usage

L'usage d'OID est pertinent dans le cadre d'applications écrites dans un langage objet, manipulant un très grand nombre d'objets reliés entre eux par un réseau d'associations complexe.

En effet si le nombre d'objets est trop grand, les objets ne peuvent tous être conservés en mémoire vive par l'application objet qui les manipule. Il est alors indispensable de les faire descendre et remonter en mémoire régulièrement. Or dans le cadre d'un traitement portant sur de très nombreux objets, la remonté en mémoire d'un objet est un point critique en terme de performance. A fortiori si l'identification de l'objet à remonter demande une interrogation complexe de la BD, à travers de nombreuses jointures. Le fait d'avoir conservé en mémoire un OID permet de retrouver et de recharger très rapidement un objet, sans avoir à le rechercher à travers des requêtes SQL comportant des jointures et donc très couteuses en performance.

Remarque : Débat

- La communauté des BD★ est plutôt contre les OID, qui rompent avec la séparation entre manipulation logique et stockage physique des données.
- La communauté objet est à l'origine de l'intégration des OID dans SQL3, en tant qu'ils sont une réponse au problème de persistance dans les applications objets.

c) Références entre enregistrements avec les OID (modèle logique)

Les OID sont une alternative aux clés étrangères pour référencer des enregistrements.

Syntaxe : Modèle logique

Dans une définition de table ou de type :

```
1 attribut =>o nom_table_d_objets
```

d) Collection imbriquée de référence d'OID

Il est possible de combiner les tables objets avec les tables imbriquées pour mobiliser des collections de références à des OID. C'est une pratique très répandue, car elle permet d'éviter les tables d'association pour les associations N:M, ce qui simplifie parfois considérablement le modèle logique.

Méthode

La création d'une collection de référence à des OID s'effectue en deux temps :

1. il faut d'abord créer un nouveau type T qui correspond à une référence à la table d'objets visée ;
2. puis créer une collection C de ce type T.

Syntaxe

```
1 type type1 : <...>
2 table1 de type1 (...)
3
4 T : <refType1 =>o table1>
5 C : collection de <T>
6
7 type2 : <... refType1:C ...>
8 table2 de type2 (...)
```

Attention : SCOPE FOR

Sous Oracle, il n'est pas possible de définir le SCOPE FOR de références à des OID imbriquées.

2. Apport des OID au passage conceptuel-logique

Objectifs

Savoir déduire un modèle logique relationnel-objet mobilisant les OID depuis un modèle conceptuel E-A ou UML.

Intégrer les apports du modèle relationnel-objet par rapport au relationnel dans ce processus.

Cette partie permet de traiter la traduction d'un modèle conceptuel UML★ ou E-A★ en modèle relationnel-objet. Le modèle E-A étendu est bien plus adapté au relationnel-objet que le modèle E-A classique.

a) Classe

Pour chaque classe (ou entité), créer un type d'objet avec les attributs de la classe (ou entité).

Créer une table d'objets de ce type pour instancier la classe (ou entité), en ajoutant les contraintes d'intégrité.

Fondamental : Table d'objet

La fait d'instancier la classe (l'entité) via une table d'objets plutôt qu'une relation classique, permet l'accès aux OID.

Complément

La déclaration d'une table objet permet également l'héritage de type, l'implémentation des méthodes.

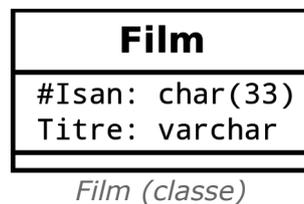
Complément : Mapping des méthodes

Si le modèle conceptuel spécifie des méthodes ou attributs dérivés, alors on peut ajouter la définition de ces méthodes sur le type d'objet créé pour chaque classe.

b) Un film classe

Question

Proposer un modèle RO correspondant au modèle UML.



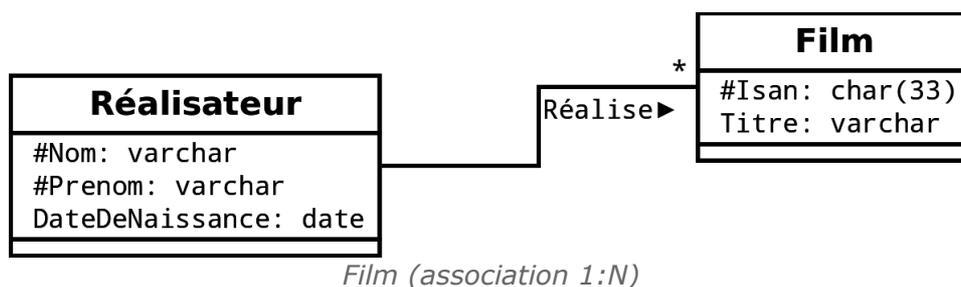
c) Association 1:N et 1:1

Les associations 1:N et 1:1 sont gérées comme en relationnel, mais on remplace la clé étrangère par une référence à un OID.

d) Un film associatif

Question

Proposer un modèle RO correspondant au modèle UML, en utilisant les OID.



e) Association N:M

Table d'association avec OID

Les associations N:M sont gérées comme en relationnel, mais on remplace les clés étrangères par des références à des OID.

Attention

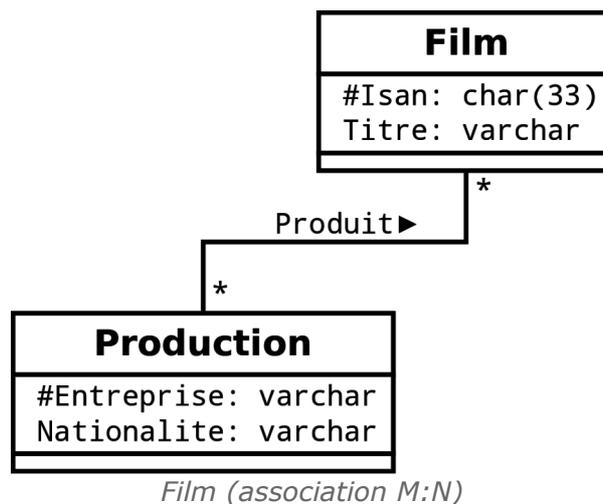
Sous Oracle on ne pourra pas créer de contrainte UNIQUE incluant un OID, donc on ne peut pas spécifier la contrainte ($\#fk1 \Rightarrow o t1, \#fk2 \Rightarrow o t2$) comme en relationnel. On acceptera en RO des tables d'association N:M sans clé primaire.

Cette impossibilité de définir une clé à partir des OID fait que l'on ne peut pas empêcher les doublons dans la table, ce qui est source d'une incohérence qui devra être gérée par ailleurs (niveau applicatif...).

f) Un film très associatif

Question

Proposer un modèle RO correspondant au modèle UML, **en utilisant les OID**.



g) Association N:M avec des collections imbriquées de références

Rappel

Les collections imbriquées (modèle et LDD)

Collection imbriquée d'OID

Il est aussi possible de gérer ces relations en utilisant une collection imbriquée de références d'OID.

Exemple : Association N:M "suit" entre "Étudiant" et "Cours"

1	Type CoursT : <...>
2	Cours de CoursT (...)
3	
4	Type RefCours : <refCours =>o Cours>
5	Type RefCoursList : collection de <RefCours>
6	Type PersonneT : <... refCours:RefCoursList ...>
7	Personne de Personne T (...)

Attention

Lorsque l'on choisit d'imbriquer la table d'association dans l'un des deux tables associées, on favorise certaines requêtes par rapport à d'autres : certaines requêtes deviennent plus facile à écrire et plus rapide à exécuter (on économise une jointure) ; d'autres au contraire deviennent plus complexes.

C'est donc l'usage des données qui oriente le choix de modélisation.

Remarque

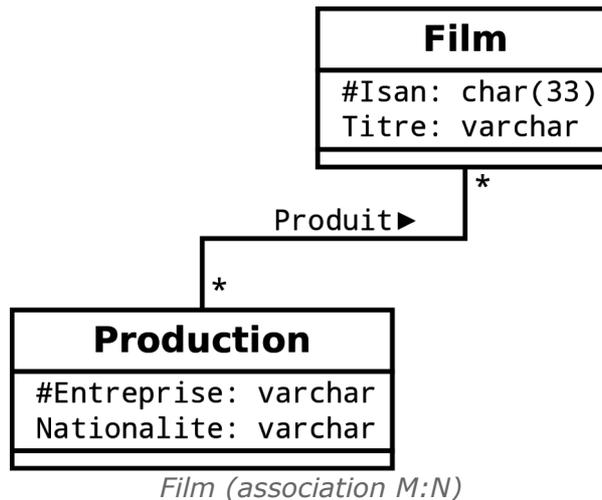
On notera que le choix d'implémentation en collection de références à des OID nous empêche dans l'implémentation sous Oracle de définir le SCOPE FOR et donc de contraindre l'intégrité référentielle à une table en particulier.

Cela ne pose pas de problème s'il y a une seule table de ce type.

h) Un film très associatif, le retour

Question

Proposer un modèle RO correspondant au modèle UML, **en utilisant les OID** et **en utilisant une table imbriquée**.



Indice :

Les films étant le centre de notre BD, c'est dans cette table que l'on imbriquera les références.

3. Méthodes et héritage dans les tables d'objets

a) Transformation des méthodes et attributs dérivés

On peut associer une méthode à une table d'objets.

La méthode se comporte comme un attribut de la table qu'il est possible de requêter, mais sa valeur n'est pas stockée, elle est recalculée dynamiquement à chaque appel.

Syntaxe

```

1 Type T : <... =duree:integer>
2 R de T(...)
  
```

b) Un film méthodique

Question

Proposer un modèle RO correspondant au modèle UML.

Film
#Isan: char(33)
Titre: varchar
DebutTournage: date
FinTournage: date
DureeTournage(): integer

Film (méthode)

c) Héritage et réutilisation de types

Définition : Héritage de type

Un type de données utilisateur peut hériter d'un autre type de données utilisateur.

Syntaxe

```

1 Type sous_type hérite de type : <
2   attributs et méthodes spécifiques
3 >
```

Remarque : Héritage de schéma de table

Pour qu'une table hérite du schéma d'une autre table, il faut définir les tables depuis des types.

L'héritage entre les types permet ainsi l'héritage entre les schémas de table.

Méthode

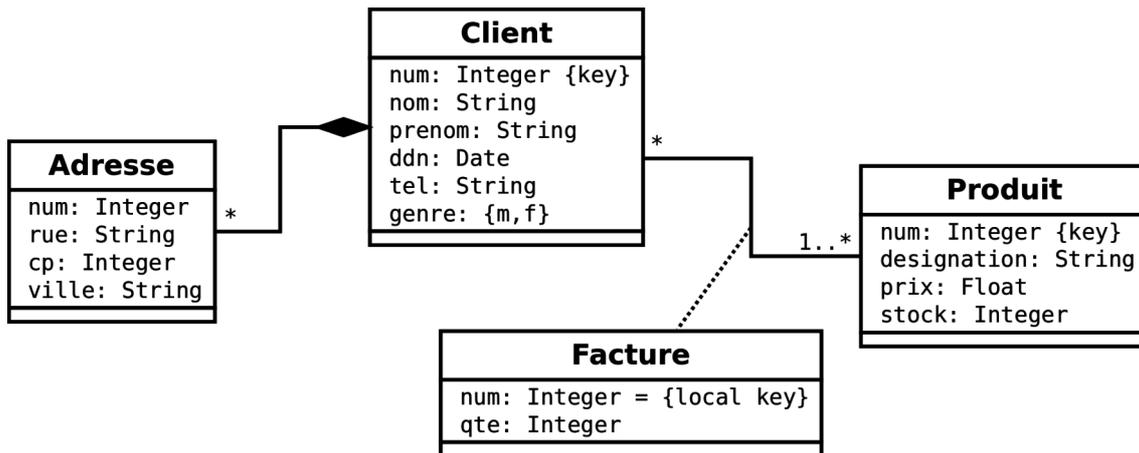
L'héritage de schéma de table n'est utile que dans les cas de transformation de l'héritage au niveau conceptuel par un héritage par les classes filles au niveau logique.

B. Exercices

1. MediaTek III

[15 min]

L'association **MediaTek** souhaite réaliser la base de données correspondant au schéma UML ci-après en utilisant un modèle relationnel-objet ne mobilisant que les tables objets (*OID*).



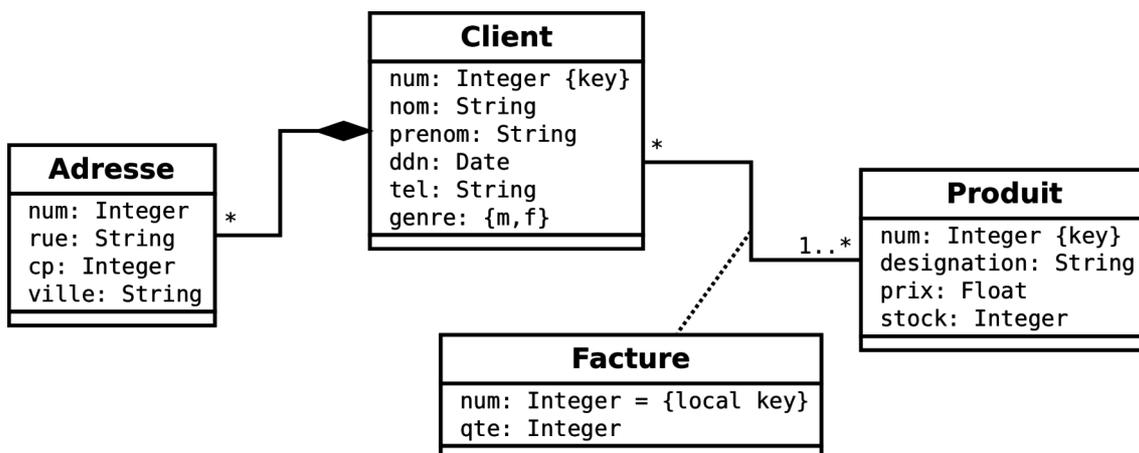
Question

Réaliser le passage vers un modèle RO, en utilisant les OID.

2. MediaTek IV

[15 min]

L'association **MediaTek** souhaite réaliser la base de données correspondant au schéma UML ci-après en utilisant un modèle relationnel-objet.



Question

Réaliser le passage vers un modèle RO, en utilisant les OID **et** l'imbrication.

On note que la principale requête du système devra afficher pour chaque client les produits qu'il a déjà acheté.

Indice :

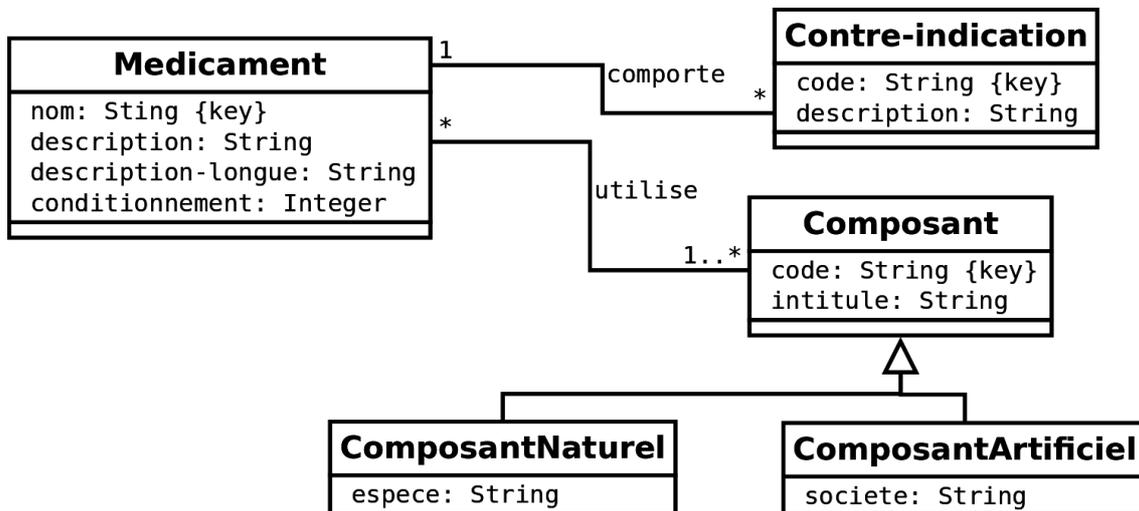
On imbrique l'association N:M dans Client.

C. Devoirs

1. Lab VI

Question

Réaliser le passage RO de ce modèle UML en utilisant les OID et références d'OID.
On utilisera ici une transformation de l'héritage par référence.



2. Usine de production II

Réaliser le passage RO correspondant à ce modèle UML.

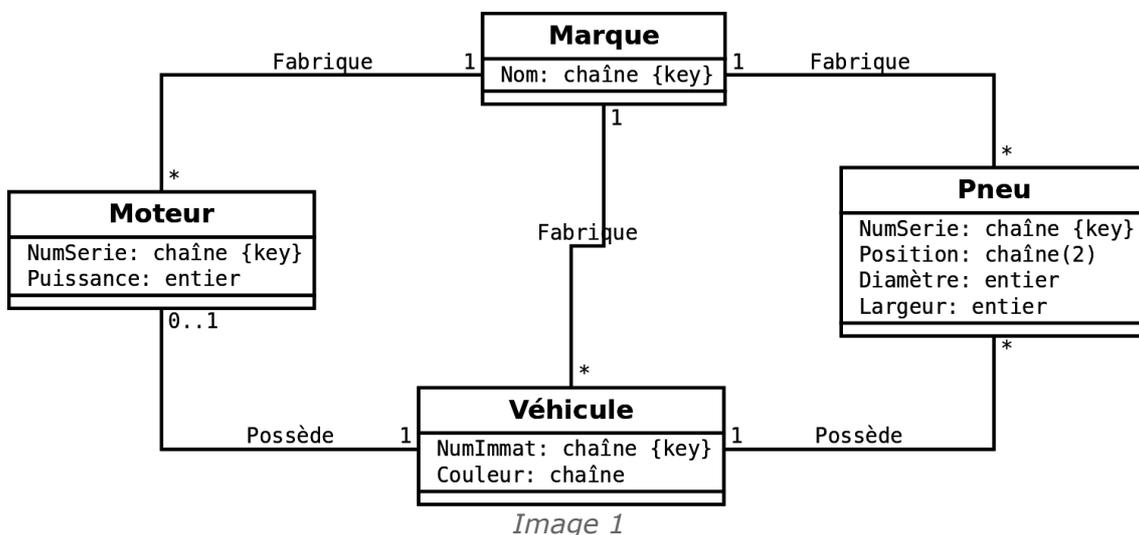


Image 1

Tables d'objets en relationnel-objet sous Oracle

VII

A. Cours

1. Tables d'objets et OID en SQL3

Objectifs

Connaître l'extension du LDD pour déclarer des tables d'objets
Connaître l'extension du LMD pour naviguer des tables d'objets

a) Création de table objet (extension au LDD SQL)

Rappel

Création de type en SQL3 sous Oracle (extension au LDD)

Syntaxe : LDD SQL3

```
1 CREATE TABLE nom_table OF nom_type (  
2   PRIMARY KEY(attribut1),  
3   attribut2 NOT NULL,  
4   UNIQUE (attribut3)  
5   FOREIGN KEY (attribut4) REFERENCES ...  
6 );
```

Il est possible, sur une table ainsi définie, de spécifier les mêmes contraintes que pour une table créée avec une clause CREATE TABLE (contraintes de table). Ces contraintes doivent être spécifiées au moment de la création de la table, et non au moment de la création du type (bien que la définition de type permet de spécifier certaines contraintes, comme NOT NULL).

Exemple : Héritage

```
1 CREATE OR REPLACE TYPE typIntervenant AS OBJECT(  
2   pknom varchar2(20),  
3   prenom varchar2(20)  
4 );  
5 /
```

```

6
7 CREATE TABLE tIntervenant OF typIntervenant (
8 PRIMARY KEY(pknom),
9 prenom NOT NULL
10 );

```

b) Références entre enregistrements avec les OID (extension au LDD SQL)

Syntaxe : LDD SQL3

```

1 CREATE TYPE type_objet AS OBJECT ...
2
3 CREATE TABLE table1 OF type_objet ...
4
5 CREATE TABLE table2 (
6 ...
7 clé_étrangère REF type_objet,
8 SCOPE FOR (clé_étrangère) IS table1,
9 );

```

Remarque : SCOPE FOR

Renforce l'intégrité référentielle en limitant la portée de la référence à une table particulière (alors que REF seul permet de pointer n'importe quel objet de ce type)

Exemple

```

1 CREATE OR REPLACE TYPE typCours AS OBJECT(
2 pkannee number(4),
3 pknum number(2),
4 titre varchar2(50),
5 type char(2),
6 refintervenant REF typIntervenant,
7 debut date,
8 MEMBER FUNCTION fin RETURN date
9 );
10 /
11
12 CREATE TABLE tCours OF typCours (
13 CHECK (pkannee>2000 and pkannee<2100),
14 type NOT NULL,
15 CHECK (type='C' or type='TD' or type='TP')),
16 refintervenant NOT NULL,
17 SCOPE FOR (refintervenant) IS tIntervenant,
18 PRIMARY KEY(pkannee, pknum)
19 );

```

c) Insertion de références par OID (INSERT)

Pour faire une référence avec un OID, il est nécessaire de récupérer l'OID correspondant à l'enregistrement que l'on veut référencer.

L'OID est accessible grâce à la syntaxe REF en SQL3.

Sous Oracle, on utilisera une procédure PL/SQL pour récupérer l'OID dans une variable avant de pouvoir l'insérer.

Syntaxe : REF

```

1 SELECT REF(alias)
2 FROM nom_table alias

```

1

```
0000280209DB703686EF7044A49F8FA67530383B36853DE7106BC74B6781275ABE5A553A5F01C0003
40000
```

Syntaxe : Insertion de données en PL/SQL

```
1 DECLARE
2   variable REF type_objet;
3 BEGIN
4   SELECT REF(alias) INTO variable
5   FROM table2 alias
6   WHERE clé_table2='valeur';
7
8   INSERT INTO tCours (champs1, ..., clé_étrangère)
9   VALUES ('valeur1', ..., variable);
10 END;
```

Exemple

```
1 DECLARE
2   refI REF typIntervenant;
3 BEGIN
4   INSERT INTO tIntervenant (pknom, prenom)
5   VALUES ('CROZAT', 'Stéphane');
6
7   SELECT REF(i) INTO refI
8   FROM tIntervenant i
9   WHERE pknom='CROZAT';
10
11  INSERT INTO tCours (pkannee, pknum, titre, type, debut, refintervenant)
12  VALUES ('2003', 1, 'Introduction','C', '01-JAN-2001', refI);
13
14  INSERT INTO tCours (pkannee, pknum, titre, type, debut, refintervenant)
15  VALUES ('2003', 2, 'Modélisation','TD', '02-JAN-2001', refI);
16 END;
17 /
```

d) Manipulation d'OID par la navigation d'objets (SELECT)

Syntaxe : Navigation d'objets

La référence par OID permet la navigation des objets sans effectuer de jointure, grâce à la syntaxe suivante :

```
1 SELECT t.reference_oid.attribut_table2
2 FROM table1 t;
```

Exemple

```
1 SELECT c.pkannee, c.pknum, c.refintervenant.pknom
2 FROM tCours c
```

```
1 PKANNEE  PKNUM  REFINTERVENANT.PKNOM
2 -----
3      2003      1  CROZAT
4      2003      2  CROZAT
```

e) RO sans fil référencé

[30 minutes]

L'on souhaite réaliser une base de données permettant de gérer tous les relais Wifi sur un site d'entreprise. Chaque relais est identifié par une coordonnée géographique (représentée par une chaîne de caractère) et fait référence à un modèle. On associe également à chaque relais son prix d'achat. Chaque modèle est décrit par une marque et un type et possède une puissance.

Question 1

Proposez une implémentation RO SQL3 sous Oracle exploitant les tables objets. Vous proposerez un MCD et un MLD préalablement pour vous aider.

Question 2

Insérer les données suivantes dans votre base de données :

- Un modèle de marque SuperWif et de type X1, puissance 25mW
- Deux relais de ce modèle respectivement aux coordonnées (48.853,2.35) et (48.978,3.01), achetés chacun 100€.

Question 3

Écrivez deux requêtes permettant de renvoyer respectivement :

- La puissance du relais 1
- La moyenne des prix des relais pour chaque modèle

2. Compléments

a) Méthodes dans les tables d'objets sous Oracle

i Méthodes de table d'objets

Définition : Méthodes de table

Si le type sur lequel s'appuie la création de la table définit des méthodes, alors les méthodes seront associées à la table (méthodes de table).

Il sera possible d'accéder à ces méthodes de la même façon que l'on accède aux attributs (projection, sélection...).

Syntaxe : Accès aux méthodes d'une table d'objets

```
1 SELECT t.m1(), t.m2() ...
2 FROM table t
3 ...
```

Attention

L'utilisation d'un alias est obligatoire pour accéder aux méthodes.

Syntaxe : Déclaration de type avec méthodes

```
1 CREATE TYPE nom_type AS OBJECT (
2   nom_attribut1 type_attribut1
3   ...
4   MEMBER FUNCTION nom_fonction1 (parametre1 IN|OUT type_parametre1, ...) RETURN
   type_fonction1
5   ...
```

```

6 );
7 /
8 CREATE TYPE BODY nom_type
9 IS
10 MEMBER FUNCTION nom_fonction1 (...) RETURN type_fonction1
11 IS
12 BEGIN
13 ...
14 END ;
15 MEMBER FUNCTION nom_fonction2 ...
16 ...
17 END ;
18 END ;
19 /

```

Exemple

```

1 CREATE TYPE typCours AS OBJECT (
2   pknum NUMBER(2),
3   debut DATE,
4   MEMBER FUNCTION fin RETURN DATE
5 );
6 /
7 CREATE TYPE BODY typCours IS
8 MEMBER FUNCTION fin RETURN DATE
9 IS
10 BEGIN
11   RETURN SELF.debut + 5;
12 END;
13 END;
14 /
15
16 CREATE TABLE tCours OF typCours (
17   pknum PRIMARY KEY
18 );
19
20 SELECT c.pknum, c.fin()
21 FROM tCours c;

```

Remarque : Type retourné par une méthode

« The datatype cannot specify a length, precision, or scale. »

http://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_5009.htm¹⁸

ii Méthodes et SELF

SELF

Lorsque l'on écrit une méthode on a généralement besoin d'utiliser les attributs propres (voire d'ailleurs les autres méthodes), de l'objet particulier que l'on est en train de manipuler.

On utilise pour cela la syntaxe SELF qui permet de faire référence à l'objet en cours.

Syntaxe : SELF

```

1 self.nom_attribut
2 self.nom_méthode(...)

```

Exemple : Total d'une facture

```

1 MEMBER FUNCTION total RETURN number
2 IS

```

18 - http://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_5009.htm

```

3   t number;
4   BEGIN
5     SELECT sum(f.qte) INTO t
6     FROM facture f
7     WHERE f.num=self.num;
8
9     RETURN t;
10  END total;

```

Remarque : SELF implicite

Dans certains cas simples, lorsqu'il n'y a aucune confusion possible, SELF peut être ignoré et le nom de l'attribut ou de la méthode directement utilisé.

Il est toutefois plus systématique et plus clair de mettre explicitement le self.

Exemple : Exemple de SELF implicite

```

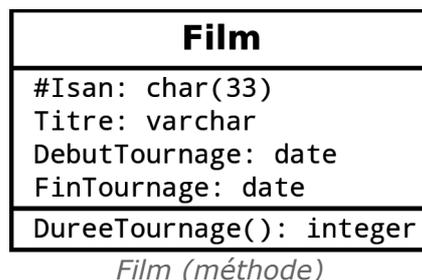
1   MEMBER FUNCTION adresse RETURN varchar2
2   IS
3   BEGIN
4     RETURN num || rue || ville;
5   END;

```

iii Un film vraiment méthodique

Question

Proposer une implémentation SQL3 sous Oracle correspondant au modèle UML. La méthode `DureeTournage` sera implémentée de façon à retourner un nombre entier de jours.



Indices :

On pourra utiliser une fonction `duree(debut,fin)` qui retourne le nombre de jours entre deux dates.

On fera un usage explicite de SELF.

b) Compléments SQL3 pour les tables objets

i Insertion d'OID sans mobiliser de PL/SQL (INSERT SELECT)

Syntaxe : Insertion de référence à des OID en SQL

```

1   INSERT INTO table1 (champs1, ..., clé_étrangère)
2     SELECT 'valeur1', ..., REF(alias)
3     FROM table2 alias
4     WHERE clé_table2='valeur';

```

ii Héritage en relationnel-objet sous Oracle

Syntaxe

```

1 CREATE TYPE sur_type AS OBJECT (
2   ...
3 ) NOT FINAL;
4 /
5 CREATE TYPE sous_type UNDER sur_type (
6   Déclarations spécifiques ou surcharges
7 ) ;
8

```

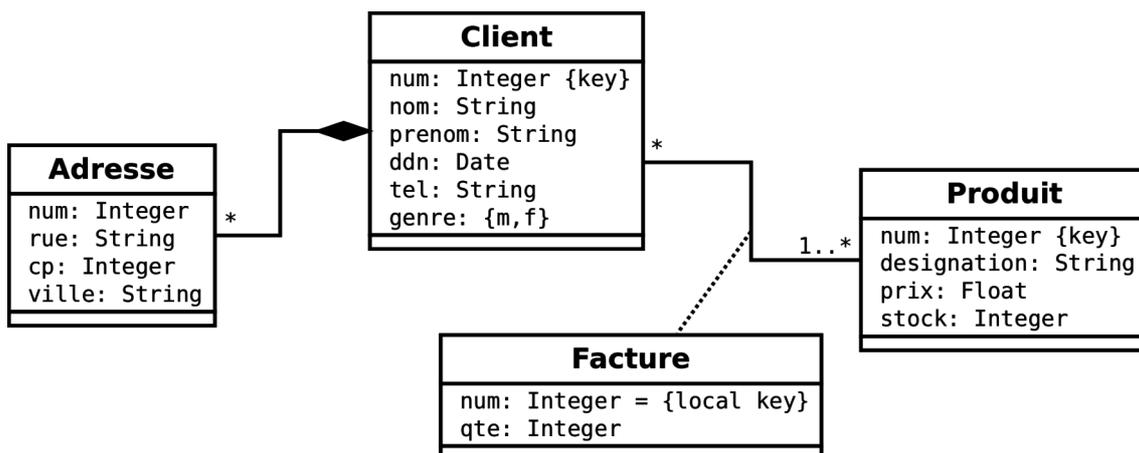
Remarque : NOT FINAL

Pour être *héritable*, un type doit être déclaré avec la clause optionnelle NOT FINAL.

B. Exercices

1. MediaTek V

[1 h]



```

1 Adresse (#num:integer, #rue:string, #cp:integer, #ville:string, #client =>
2   Client)
3
4 Type TClient <num:integer, nom:string, prenom:string, ddn:date, tel:string,
5   genre: {'m', 'f'}>
6 Client de TClient (#num)
7
8 Type TProduit <num:integer, designation:string, prix:float, stock:integer>
9 Produit de TProduit (#num)
10
11 Facture (#num:integer, #client => Client, #produit => Produit, qte:integer)

```

Question 1

Réalisez l'implémentation SQL3 sous Oracle.

Alimenter une base RO

Une fois la base créée, insérer des données pour l'alimenter.

On négligera la table Adresse pour la suite de l'exercice (aucun de nos clients n'aura d'adresse).

Question 2

Initialiser les tables client et produit avec les données de votre choix (au moins deux clients et deux produits).

Indice :

Effectuez l'insertion comme en relationnel.

Question 3

Initialiser les factures avec les données de votre choix (deux factures de deux lignes chacune au moins).

Indices :

*Récupérer les **OID** pour insérer les références aux produits et clients au sein d'une procédure PL/SQL.*

Insertion de références par OID (INSERT)

Blocs PL/SQL : Procédure, fonction, bloc anonyme

```

1 DECLARE
2   oidclient1 REF TClient;
3   ...
4
5 BEGIN
6   SELECT REF(c) INTO oidclient1
7   FROM Client c
8   WHERE c.num=...;
9
10  ...
11
12  INSERT INTO Facture (num, client, produit, qte)
13  VALUES (1,oidclient1, ..., 3);
14
15  ...
16
17 END;
18 /

```

Questions par navigation d'OID

Vous allez à présent expérimenter la manipulation des OID pour **naviguer** d'enregistrement en enregistrement, sans utiliser de jointure.

Question 4

Écrivez une requête permettant d'afficher la liste des factures existantes, avec le nom et le prénom du client.

Indice :

1	NUM CLIENT.NOM	CLIENT.PRENOM
2	-----	-----
3	1 Colomb	Christophe
4	1 Morin	Bernard

Question 5

Écrivez une requête permettant d'afficher le montant total de chaque facture, en rappelant le numéro du client pour chaque facture.

Indice :

1	NUM CLI	TOTAL
2	---	----
3	1	1 169.1
4	1	2 319.3

Question 6

Écrivez une requête permettant d'afficher pour chaque client (num et nom) les produits (num et désignation) qu'il a déjà acheté, avec la quantité correspondante.

Indice :

1	CLIENT.NUM	CLIENT.NOM	PRODUIT.NUM	PRODUIT.DESIGNATION	QTE
2	-----	-----	-----	-----	-----
3		1 Colomb	1	The Matrix	3
4		1 Colomb	2	The Hobbit	2
5		2 Morin	1	The Matrix	1
6		2 Morin	2	The Hobbit	6

2. Des voitures et des hommes

[45 minutes]

Soit le diagramme de classe UML suivant :

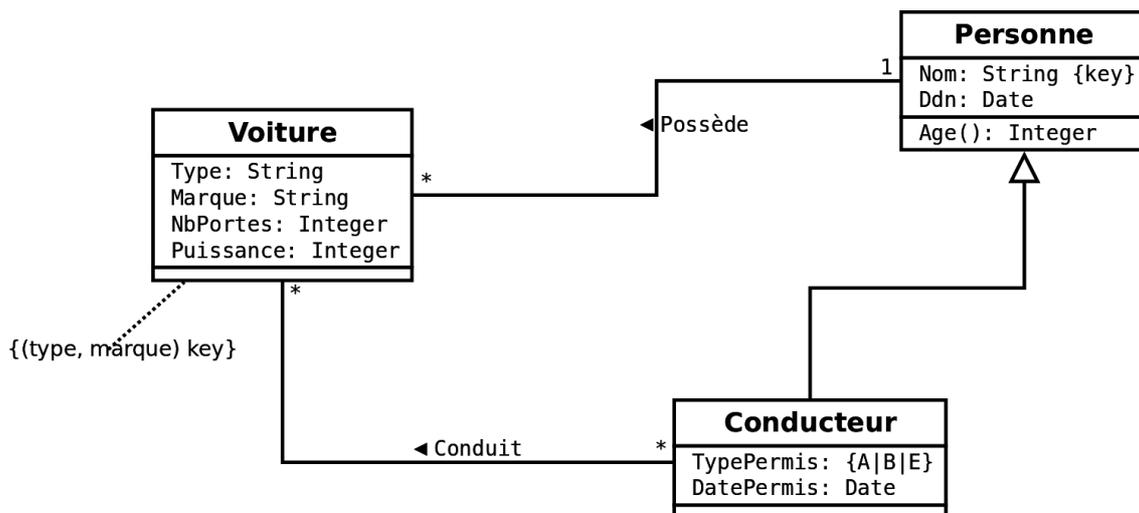


Image 2

Question 1

A partir de ce modèle conceptuel établissez un modèle logique en relationnel-objet.

On utilisera des tables d'objets et les OID pour effectuer les références (en revanche, on n'utilisera pas le modèle imbriqué).

Question 2

Proposer une implémentation sous Oracle de votre modèle logique (sans implémenter les méthodes et sans utiliser l'héritage de type).

Question 3

Implémenter la méthode Age() pour Personne et pour Conducteur.

Indice :

```
1 trunc(months_between(SYSDATE, ddn) / 12
```

Question complémentaire (héritage de type)

L'amélioration proposée ci-après évite notamment la double déclaration de la méthode `age`.

Question 4

Proposer une solution mobilisant l'héritage de type afin de factoriser la déclaration des attributs `nom` et `ddn` et la méthode `age`.

3. Des voitures et des hommes de collection

[45 minutes]

Soit le diagramme de classe UML suivant :

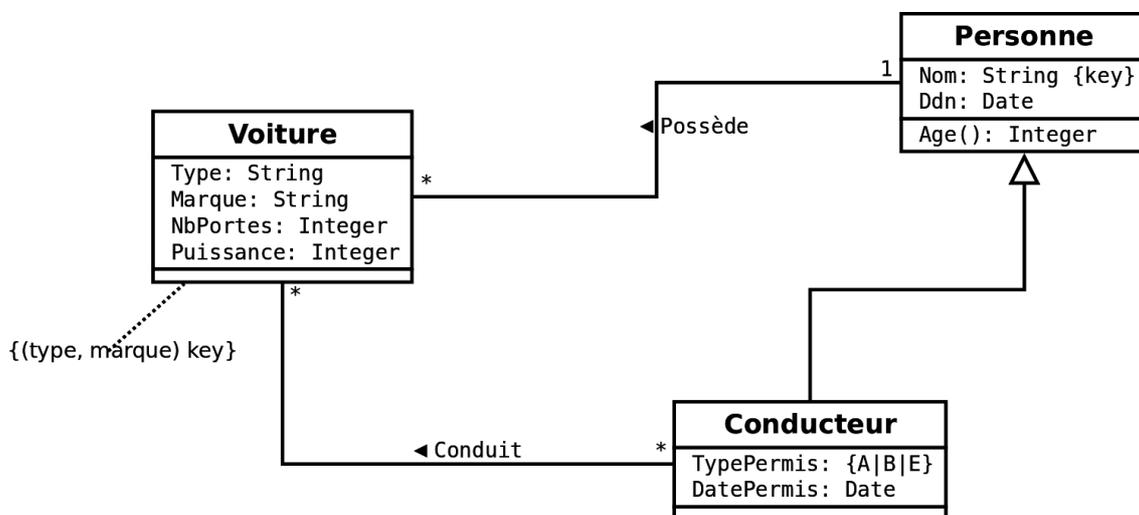


Image 3

Question 1

A partir de ce modèle conceptuel établissez un modèle logique en relationnel-objet.

On utilisera des tables d'objets et les OID pour effectuer les références, ainsi qu'une table imbriquée pour gérer l'association N:M comme une collection de référence à des OID.

On privilégiera la table `Voiture`, qui sera au centre des requêtes posées à la BD.

Question 2

Proposer une implémentation sous Oracle de votre modèle logique (sans implémenter les méthodes et sans utiliser l'héritage de type).

Question 3

Insérer une voiture, un propriétaire et deux conducteurs dans la base de données.

Question 4

Afficher toutes les voitures avec les personnes qui les possèdent et les conduisent.

C. Devoirs

1. Zoologie

[45 minutes]

Un zoologue souhaite réaliser une base de données pour l'étude des chaînes alimentaires.

Il souhaite étudier deux populations d'animaux : les carnivores et les herbivores, sachant que les carnivores ne mangent **que** des herbivores et que les herbivores, évidemment, ne mangent pas d'animaux. La base gère donc des espèces d'animaux, identifiées par leur nom (Lion, Biches, etc.). Les carnivores et les herbivores sont étudiés selon deux paramètres communs : taille moyenne et poids moyen.

Pour les herbivores, on étudie en plus le poids moyen de végétaux mangé par jour. Bien entendu le zoologue souhaite connaître également le poids moyen mangé par jour pour chaque espèce de carnivore et pour chaque espèce d'herbivore qu'il mange. Il souhaite enfin pouvoir obtenir très facilement le poids total mangé par un carnivore par jour (donc en sommant le poids de chaque herbivore mangé par jour). On fera apparaître cette opération sur le modèle.

Question 1

Réalisez le modèle conceptuel du problème posé.

Question 2

Réalisez le modèle logique **relationnel-objet** du problème (on utilisera les OID pour toutes les tables référencés).

On ajoutera la méthode *poidsMangéParJourTotal()* à la table d'objets *Carnivore*.

Question 3

Implémentez le modèle relationnel-objet en SQL3 sous Oracle (sans implémenter de méthode).

Question 4

Implémentez la méthode permettant de calculer le poids total d'herbivore mangé par un carnivore par jour.

Question 5

Écrivez une requête qui permet de retourner tous les carnivores, triés par leur poids, avec le poids total d'herbivore qu'ils mangent par jour.

Question 6

Proposez et implémentez une solution d'optimisation pour améliorer les performances de la requête précédente. Justifiez votre choix et expliquez, éventuellement à l'aide d'un petit schéma ou d'un exemple, pourquoi l'exécution sera améliorée.

Modélisation logique arborescente en XML

VIII

A. Cours

1. Introduction à XML

a) Définition du XML

Définition : XML

L'eXtensible Markup Language XML [[w_w3c.org/XML](http://w3c.org/XML)] est un **méta-langage** permettant de définir des langages à **balises**.

Il standardisé comme une recommandation par le W3C★ depuis **1998**.

Exemple

- LaTeX et HTML sont des langages à balises (mais ce ne sont pas des langages XML)
- XHTML est un langage XML

Fondamental

En tant que méta-langage XML sert à définir des formats informatiques, c'est à dire des façons de représenter de l'information.

Les bonnes caractéristiques d'XML - non-ambiguïté, lisibilité, passivité - en font un très bon candidat pour de très nombreux usages, il est donc utilisé dans des secteurs très variés de l'informatique.

On distingue généralement deux grandes catégories d'utilisation : les langages orientés données et les langages orientés documents.

Complément : Versions de XML

La version historique de XML est la version 1.0, qui reste aujourd'hui la plus largement utilisée. Il existe également une version 1.1 de XML, qui propose des différences mineures et nécessaires uniquement dans des contextes particuliers. Cette nouvelle version a été nécessaire pour des raisons de compatibilité des outils existants. Les évolutions principales de XML 1.1 sont de nouveaux caractères permis pour les noms d'éléments (pour suivre l'évolution

d'Unicode depuis 1998), de nouvelles conventions pour les caractères de fin de ligne (pour la compatibilité avec des ordinateurs *main frame*) et de nouveaux caractères de contrôle dans le contenu.

Complément : Voir aussi

XML : Un langage à balise

XML : un méta-langage

Langages XML orientés données

Langages XML orientés documents

Caractéristiques recherchées pour un format XML

b) Document bien formé

Définition

Un document est dit bien formé lorsqu'il respecte les règles syntaxiques du XML★.

Fondamental

XML★ ne pose pas de sémantique à priori mais uniquement des règles syntaxiques.

Syntaxe

Les règles syntaxiques à respecter sont :

- Il n'existe qu'un seul élément père par document, cet élément contenant tous les autres. Il est également appelé **racine**.
- Tout élément fils est inclus complètement dans son père (pas d'éléments croisés).

Attention

XML ne définit pas le comportement ni la manière dont doit être traité un document, mais uniquement un format.

Complément

Balise

c) Exemple : Un document XML

Exemple : Un mail

```

1 <?xml version='1.0' encoding='iso-8859-1'?>
2 <mail>
3   <de>stephane.crozat@utc.fr</de>
4   <a>fabrice.issac@utc.fr</a>
5   <objet>Demande d'information</objet>
6   <date>01-03-2001</date>
7   <corps>
8     <paragraphe>Bonjour Fabrice,</paragraphe>
9     <paragraphe>Peux tu m'expliquer ce qu'est le langage XML ?</paragraphe>
10    <paragraphe>Il me faudrait en effet ton avis éclairé sur le
    sujet.</paragraphe>
11    <paragraphe>J'attends ta réponse, à bientôt</paragraphe>
12  </corps>
13 </mail>

```

d) Exercice

Compléter les trous avec les balises **fermantes** adéquates.

<?xml version="1.0"?>

```

<document>
  <entete>
    <titre>Document à corriger</titre>
    <auteur>Stéphane Crozat [redacted]</auteur>
    <date>01-03-01 [redacted]</date>
    <version numero="1"/>
    [redacted]
  <corps>
    <division titre="Première partie">
      <paragraphe>Ce texte doit être <important>corrige [redacted]</paragraphe>
      <paragraphe>Le contenu est sans importance ici</paragraphe>
    </division>
    <division titre="Seconde partie">
      <paragraphe>Ai-je le droit de mettre du texte ici ?</paragraphe>
      [redacted]
      [redacted]
      [redacted]

```

e) Langages XML orientés données

Définition

Ils permettent d'enregistrer et de transporter des données informatiques structurées (comme par exemple des données gérées par des bases de données) selon des formats ouverts (c'est à dire dont on connaît la syntaxe) et faciles à manipuler (les structures arborescentes XML étant plus riches que des fichiers à plat par exemple).

Les langages XML orientés données servent surtout à l'échange ou la sérialisation des données des programmes informatiques.

Remarque

L'utilisation d'XML est en fait ici assez accessoire, d'autres formalismes s'y substituent sans difficulté, souvent moins explicites pour la manipulation humaine et souvent plus performants pour la manipulation informatique : CSV, JSON, ... (voir par exemple : <http://www.xul.fr/ajax-format-json.html>¹⁹ pour une comparaison JSON / XML).

Remarque : Format verbeux

XML est en général plus verbeux que ses alternatives (il contient plus de caractères), c'est pourquoi il est plus explicite pour un humain (ce qui n'est pas toujours utile), et moins performant informatiquement (ce qui n'est pas toujours un problème).

Exemple : Formats XML orientés données standards

- MathML, ChemML, ...
- ATOM, RSS
- Dublin Core
- RDF, OWL
- SVG
- ...

19 - <http://www.xul.fr/ajax-format-json.html>

Exemple : Formats XML orientés données locaux

XML est utilisé pour de nombreux langages orientés données locaux, en fait chaque fois qu'un format XML est défini pour les besoins spécifique d'un programme.

```

1 <myVector>
2   <x>5</x>
3   <y>19</y>
4 </myVector>

```

Complément : Langages XML de programmation

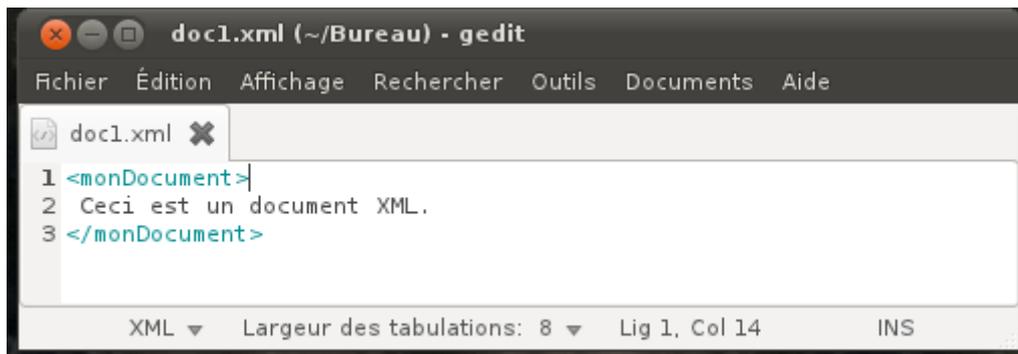
Le formalisme XML est également utilisé pour définir des langages de programmation, à l'instar du C ou du Java. Les langages de programmation écrits en XML sont généralement à vocation déclarative et adressent le plus souvent des traitements eux mêmes liés à XML.

XSL-XSLT est un exemple de langage de programmation écrit en XML. On peut également citer par exemple le langage de script ANT (<http://ant.apache.org/>²⁰) ou XUL pour la réalisation d'IHM (<http://www.xul.fr/>²¹)

f) Outillage XML

Fondamental

Un éditeur de texte suffit pour faire du XML.



Document XML créé avec un simple éditeur de texte (gedit)

Parseurs

Un *parser* (ou parseur) XML est un programme capable d'analyser un document XML afin de :

- vérifier qu'il est bien formé
- éventuellement vérifier sa validité par rapport à un schéma (DTD, W3C Schema, RelaxNG)
- éventuellement en fournir une représentation mémoire permettant son traitement (SAX, DOM)

Par exemple Xerces est un parseur Java (<http://xerces.apache.org/xerces2-j/>²²). Tous les langages de programmation proposent aujourd'hui des parseurs XML.

Éditeur validant

Un éditeur validant est un éditeur spécialisé dans l'écriture du XML (on parle simplement d'éditeur XML), permettant de :

- vérifier dynamiquement que le fichier est bien formé,
- de charger des schémas pour vérifier dynamiquement la validité,

20 - <http://ant.apache.org/>

21 - <http://www.xul.fr/>

22 - <http://xerces.apache.org/xerces2-j/>

- de proposer des fonctions d'auto-complétion,
- ...

Éditeurs XML orientés développeurs

Les éditeurs spécialisés orientés développeurs sont des IDE★ permettant l'écriture :

- des schémas,
- des transformations,
- des fichiers XML de test,
- ...

Ils sont en général peu adaptés à la production des documents XML par les utilisateurs finaux (rédacteurs).

On peut citer par exemple :

- oXygen (<http://www.oxygenxml.com/>²³), un produit commercial complet accessible à moins de 100€ pour les usages non commerciaux, multi-plateformes (Linux, Mac, Windows) ;
- XML Spy, disponible uniquement sous Windows
- ...

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
3     "http://www.docbook.org/xml/4.4/docbookx.dtd">
4 <article>
5     <title>Outillage XML</title>
6     <sect1>
7         <title>Parseurs</title>
8         <para>Un <foreignphrase>parser</foreignphrase>(ou parseur) XML est un programme c
9         <itemizedlist>
10            <listitem><para>vérifier qu'il est bien formé</para></listitem>
11            <listitem><para>éventuellement vérifier sa validité</para></listitem>
12        </itemizedlist>
13    </sect1>
14 </article>

```

Exemple de fichier XML dans l'éditeur oXygen

Éditeurs XML orientés rédacteurs

Les éditeurs XML orientés rédacteurs prennent en général en entrée un schéma, des fichiers permettant de configurer l'éditeur, des programmes de transformation, et offre un environnement dédié à l'écriture et la publication de document XML.

On peut citer :

- Jaxe, libre et multi-plateformes (<http://jaxe.sourceforge.net/fr/>²⁴)
- Adobe Framemaker, sous Windows (<http://www.adobe.com/products/framemaker.html>²⁵)
- Arbortext, sous Windows (<http://www.ptc.com/products/arbortext/>²⁶)
- XMetal, sous Windows (<http://na.justsystems.com/content-xmetal/>²⁷)

23 - <http://www.oxygenxml.com/>

24 - <http://jaxe.sourceforge.net/fr/>

25 - <http://www.adobe.com/products/framemaker.html>

26 - <http://www.ptc.com/products/arbortext/>

27 - <http://na.justsystems.com/content-xmetal>

- Scenari, chaîne éditoriale complète libre et multi-plateformes (<http://scenari.org/28>)

Complément : Voir aussi

Définition des chaînes éditoriales XML

Définition de Scenari

2. Syntaxe de base XML

a) Balise

Définition : Balise

Les balises sont les composants fondamentaux permettant l'écriture de documents XML★. Elles se distinguent du contenu en utilisant les caractères <, > et /. Elles n'ont pas de présentation ou de signification définie par le langage mais elles auront un sens pour les applications et/ou le lecteur.

Syntaxe

Il existe trois types de balises :

- balise d'ouverture : <nom_balise>
- balise de fermeture : </nom_balise>
- balise vide : <nom_balise/>

Exemple : Exemple de balise XML

```
1 <adresse>12, rue de Paris</adresse>
```

b) Élément

Définition : Élément

Un élément XML★ est un extrait d'un fichier XML comprenant :

- une balise ouvrante
- une balise fermante
- le contenu compris entre les deux balises, qui peut être du **texte** et/ou d'autres **éléments**, ou **rien** (élément vide)

Le nom d'un élément peut être composé de tout caractère alphanumérique plus _, - et .. De plus, ils doivent commencer par un caractère alphabétique ou _ et ceux commençant par la chaîne xml sont réservés (que ce soit en minuscules, majuscules ou un mélange des deux).

Attention : Case-sensitive

XML★ différencie les majuscules des minuscules, il faut donc respecter la casse.

Attention

Deux éléments ne peuvent pas avoir un contenu croisé : <nom1> ... <nom2> ... </nom1> ... </nom2> est interdit.

Définition : Élément vide

On appelle élément vide un élément qui ne comporte rien entre sa balise ouvrante et sa balise fermante.

28 - <http://scenari.org/>

Syntaxe : Élément vide

Les syntaxes `<element></element>` et `<element/>` sont strictement équivalentes.

c) Attribut

Définition

Un attribut est une information supplémentaire attachée à un élément, on parle de métadonnée.

Syntaxe

Les attributs d'un élément sont formés d'une suite d'affectations séparées par des espaces : `attribut1='valeur' attribut2='valeur' ...`

Ils sont ajoutés à la balise ouvrante ou à une balise vide (jamais à une balise fermante) :

- `<nom_element [attributs]>`
- `<nom_element [attributs]/>`

La valeur est indiquée entre apostrophes ou guillemets (au choix, mais pas de mélange des deux) :

- `attribut1='valeur' OU`
- `attribut1="valeur"`

Méthode

Utilisez des apostrophes si la valeur de l'attribut inclut des guillemets et vice et versa.

Attention

Un élément ne peut pas contenir deux attributs ayant le même nom.

Syntaxe

Le nom d'un attribut est soumis aux mêmes contraintes que les noms d'éléments.

La valeur de l'attribut quant à elle peut contenir tout caractère à l'exception de `^`, `%` et `&`.

Remarque : Équivalence attribut / élément

Un attribut peut toujours être représenté alternativement par un élément fils de l'élément qu'il caractérise, avec une signification du même ordre :

- `<element attribut="x"/>` et
- `<element><attribut>x</attribut><element>` sont similaires.

Il est donc tout à fait possible de faire du XML sans utiliser d'attribut.

Méthode : Usage des attributs

On utilise généralement les attributs :

- Pour différencier le contenu destiné à être affiché dans le document lisible des métadonnées qui ne le seront pas (version, date de création, ...)
- Pour simplifier l'écriture du document
- Pour ajouter des identifiants et des références

d) Structure générale d'un fichier XML

Syntaxe

Un document XML★ est constitué de :

- Un **prologue**

Il est facultatif et comprend une déclaration XML★, indiquant la version du langage XML utilisé et le codage des caractères dans le document. Chacune de ces informations est optionnelle mais leur ordre est obligatoire

```
<?xml version="numéro de version" encoding="encodage des caractères"?>
```

- Un **arbre d'éléments** contenant au moins un élément (l'élément racine)

Exemple : Prologue

```
1 <?xml version="1.0" encoding="UTF-8"?>
```

Indique que le document est codé en utilisant un langage XML★ de version 1.0, avec des caractères codés selon la norme UTF-8.

Exemple : Arbre d'éléments

```
1 <lettre>
2   <expediteur>moi</expediteur>
3 </lettre>
```

e) Exemple de fichier XML : un courriel

Exemple : Un courriel imprimé

Date: Mar, 28 Oct 2003 14:01:01 +0100 (CET)

De : Marcel <marcel@ici.fr>

A : Robert <robert@labas.fr>

Sujet: Hirondelle

Salut,

Pourrais-tu m'indiquer quelle est la vitesse de vol d'une hirondelle transportant une noix de coco ?

A très bientôt,

Marcel

Représentation possible de ce message en XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <email>
3   <entete>
4     <date type="JJMMAAAA">28102003</date>
5     <heure type="24" local="(GMT+01 :00)">14:01:01</heure>
6     <expediteur>
7       <adresse mail="marcel@ici.fr">Marcel</adresse>
8     </expediteur>
9     <recepteur>
10      <adresse mail="robert@labas.fr">Robert</adresse>
11    </recepteur>
12    < sujet>Hirondelle</ sujet>
13  </entete>
14  <corps>
15    <salutation>Salut,</salutation>
16    <paragraphe>Pourrais-tu m'indiquer quelle est la vitesse de vol d'une
17  hirondelle
18    transportant une noix de coco ?</paragraphe>
19    <politesse>A très bientôt,</politesse>
20    <signature>Marcel</signature>
21  </corps>
22 </email>
```

3. Introduction aux schémas XML

a) Notion de document valide

Définition : Schéma

Un schéma est une description de la structure que doit respecter un document lui faisant référence, c'est à dire qu'il établit la liste des éléments XML autorisés (avec leurs attributs), ainsi que l'agencement possible de ces éléments.

On parle aussi de **grammaire**, au sens où le schéma définit l'enchaînement autorisé des balises et vient en **complément de la syntaxe XML** (qui elle est indépendante d'un schéma particulier).

Définition : Document valide

Un document XML★ bien formé est dit valide pour un schéma donné s'il respecte les règles structurelles imposées par ce schéma.

Ce contrôle de la structure permet :

- De s'assurer l'homogénéité structurelle des documents de même type.
- Le traitement automatique d'un ensemble de documents de même type (mise en forme, stockage, extraction d'informations...).
- La création de formats standard et leur respect.

Exemple : Exemples de langages de schéma

Il existe plusieurs langages de définition schéma, mais les trois principaux sont :

- Document Type Définition (W3C) : Un langage hérité de SGML qui fait partie du standard XML
- W3C XML Schema (W3C) : Une alternative aux DTD destiné à moderniser et compléter ce langage historique
- Relax NG (OASIS, ISO) : Une autre alternative, compromis entre W3C XML Schema et DTD

b) Document Type Definition

Le formalisme de définition de schéma DTD est le premier qui a été introduit dès la première version du standard XML. Il est en fait intégré au standard W3C de XML.

Il est directement hérité de la norme SGML.

Les DTDs utilisent un langage spécifique (non XML) pour définir les règles structurelles. Un fichier de DTD peut contenir principalement deux types de déclarations :

- **des déclarations d'éléments**,
indiquent les éléments pouvant être inclus dans un document et l'organisation du contenu de chaque élément (éléments fils ou texte).
- **des déclarations d'attributs**,
définissent les attributs pouvant être associés à un élément ainsi que leur type.

Exemple : Exemple de DTD

```
1 <!ELEMENT document (paragraphe+)>
2 <!ATTLIST document type CDATA #REQUIRED>
3 <!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de document XML valide

```
1 <?xml version='1.0' encoding='iso-8859-1'?>
```

```

2 <!DOCTYPE document SYSTEM "document.dtd">
3 <document type='memo'>
4   <paragraphe>Lorem ipsum dolor sit amet.</paragraphe>
5   <paragraphe>Consectetur adipiscing elit.</paragraphe>
6   <paragraphe>Sed do eiusmod tempor.</paragraphe>
7 </document>

```

c) Exercice

Le fichier *file.xml* n'est pas valide par rapport à la DTD *schema.dtd*. Sélectionnez les éléments causes de cette non-validité.

```

1 <?xml version="1.0"?>
2 <!--file.xml-->
3 <!DOCTYPE papier SYSTEM "schema.dtd">
4 <papier>
5   <titre>Réinterroger les structures documentaires</titre>
6   <auteur>Stéphane Crozat</auteur>
7   <auteur>Bruno Bachimont</auteur>
8   <resume>Nous proposons dans cet article d'aborder ...</resume>
9   <abstract>In this paper we define...</abstract>
10 </papier>

```

```

1 <!-- schema.dtd-->
2 <!ELEMENT papier (titre, sousTitre?, auteur, resume)>
3 <!ELEMENT titre (#PCDATA)>
4 <!ELEMENT sousTitre (#PCDATA)>
5 <!ELEMENT auteur (#PCDATA)>
6 <!ELEMENT resume (#PCDATA)>

```

- 1 - abstract
- 2 - resume
- 3 - auteur
- 4 - sousTitre
- 5 - titre

Éléments correctement spécifiés

Éléments incorrectement spécifiés

d) Relax NG : Syntaxe XML

Syntaxe : Syntaxe générale

```

1 <grammar xmlns="http://relaxng.org/ns/structure/1.0">
2   <start>
3     <element name="...">
4       ...
5     </element>
6   </start>
7 </grammar>

```

Syntaxe : Imbrication

```

1 <element name="">
2   <element name="">
3     ...
4   </element>
5 </element>

```

Syntaxe : Séquentialité

```

1 <element name="">
2   <element name=""> ... </element>
3   <element name=""> ... </element>
4   ...
5 </element>

```

Syntaxe : Attributs

```

1 <element name="">
2   <attribute name="">
3   ...
4 </attribute>
5 </element>

```

Syntaxe : Nœuds texte

```

1 <element name="">
2   <text/>
3 </element>

```

Syntaxe : Cardinalité

```

1 <element name="">
2   <zeroOrMore>
3     <element name=""> ... </element>
4   </zeroOrMore>
5   <oneOrMore>
6     <element name=""> ... </element>
7   </oneOrMore>
8   ...
9 </element>

```

Syntaxe : Optionalité

```

1 <element name="">
2   <optional>
3     <element name=""> ... </element>
4   </optional>
5   ...
6 </element>

```

Syntaxe : Alternative

```

1 <element name="">
2   <choice>
3     <element name=""> ... </element>
4     <element name=""> ... </element>
5     <element name=""> ... </element>
6   </choice>
7   ...
8 </element>

```

Syntaxe : Énumération

```

1 <attribute name="">
2   <choice>
3     <value>a</value>

```

```

4         <value>b</value>
5         <value>c</value>
6     </choice>
7 </attribute>

```

Complément

<http://www.oasis-open.org/committees/relax-ng/tutorial.html>²⁹

<http://xmlfr.org/oasis/committees/relax-ng/tutorial-20011203-fr.html>³⁰

e) Types de données

Syntaxe

Ajouter l'attribut `datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"` à la racine `grammar`.

```

1 <grammar
2     xmlns="http://relaxng.org/ns/structure/1.0"
3     datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
4     ...
5     <element name="...">
6         <data type="...">
7     </element>
8 </grammar>

```

Définition : Types primitifs

- string
- boolean
- decimal
- float, double
- date, dateTime, duration, time, gYearMonth, gYear, gMonthDay, gDay, gMonth
- hexBinary, base64Binary
- anyURI
- QName, NOTATION
- Types hérités des DTD : ID, IDREF, IDREFS...

Complément : Spécification des primitive datatypes

<http://www.w3.org/TR/xmlschema-2/>³¹

Facette

Paramètre de spécialisation des types primitifs.

Exemple : Type string

Facettes :

- length : longueur de la chaîne
- pattern : expression régulière permettant de valider la chaîne par rapport au patron (définition en intention)
- enumeration : liste de valeurs autorisées (définition en extension)
- ...

29 - <http://www.oasis-open.org/committees/relax-ng/tutorial.html>

30 - <http://xmlfr.org/oasis/committees/relax-ng/tutorial-20011203-fr.html>

31 - <http://www.w3.org/TR/xmlschema-2/>

Définition : Built-in datatypes

Dérivés des types primitifs.

Par exemple :

- integer, dérivé de decimal
- normalizedString, dérivé de string
- language, dérivé de string
- ID, IDREF

Exemple : RelaxNG XML

```

1 <grammar xmlns="http://relaxng.org/ns/structure/1.0">
2   <start>
3     <element name="mail" datatypeLibrary="http://www.w3.org/2001/XMLSchema-
  datatypes">
4       <data type="string">
5         <param name="pattern">([ ^ ])+@([ ^ ])+.([ ^ ])+</param>
6       </data>
7     </element>
8   </start>
9 </grammar>

```

Exemple : RelaxNG compact

```

1 datatypes xsd="http://www.w3.org/2001/XMLSchema-datatypes"
2 start = element age {xsd:decimal {maxInclusive="100"}}

```

Complément

<http://www.xml.dvint.com/docs/SchemaDataTypesQR-2.pdf>³²

Complément

Guidelines for using W3C XML Schema Datatypes with RELAX NG :
<http://relaxng.org/xsd.html>³³

f) Présentation de oXygen XML Editor

Oxygen XML Editor est un environnement de développement XML.

Il permet notamment de

- créer des documents XML bien formés,
- créer des schémas et valider des documents XML
- créer des transformations XSLT et les exécuter sur des documents XML

Toutes ces fonctions sont disponibles à partir du menu : Document > Document XML.

Il permet également de tester des expressions XPath sur un document XML ouvert dans l'éditeur.

Installation

Il peut être téléchargé ici : <http://www.oxygenxml.com>³⁴ et il est possible de bénéficier d'une licence d'essai de 30 jours.

Les anciennes versions d'Oxygen peuvent être téléchargées à l'adresse : http://www.oxygenxml.com/software_archive_editor.html³⁵

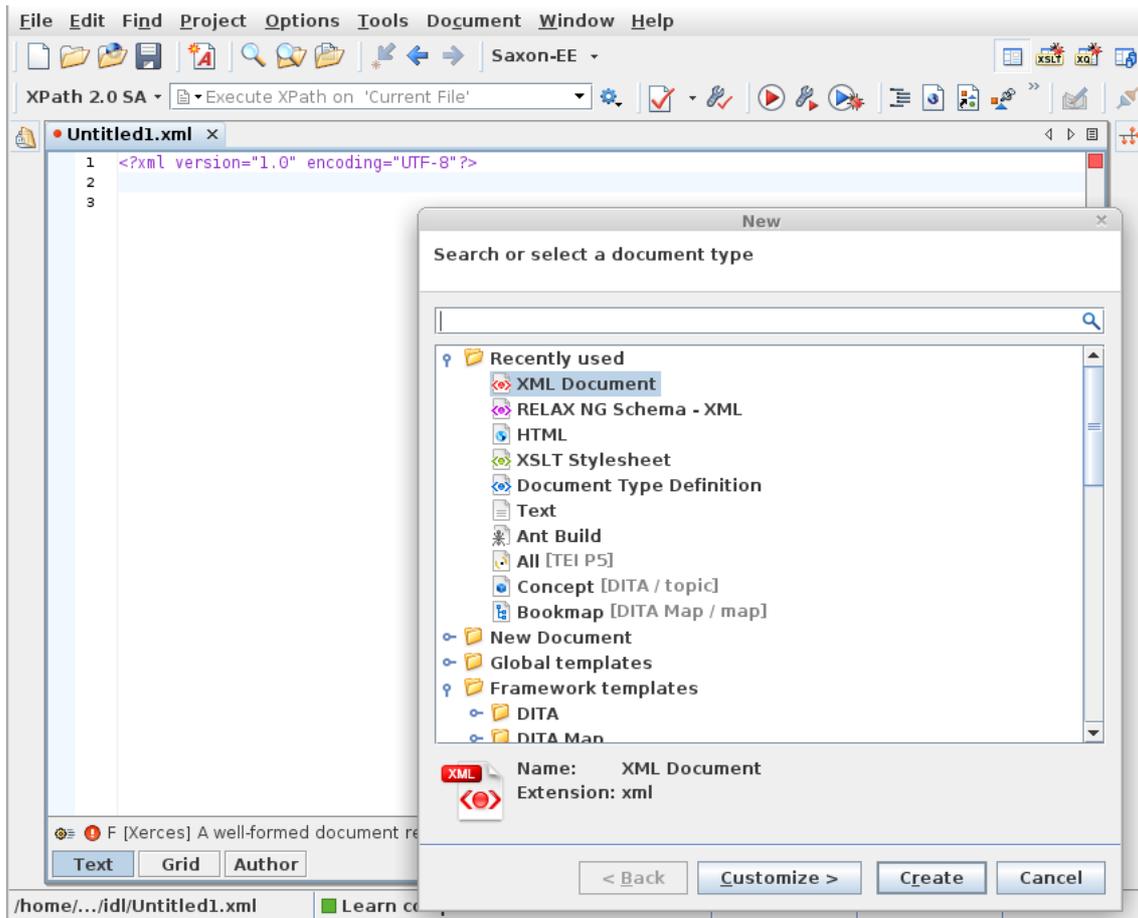
32 - <http://www.xml.dvint.com/docs/SchemaDataTypesQR-2.pdf>

33 - <http://relaxng.org/xsd.html>

34 - <http://www.oxygenxml.com/>

35 - http://www.oxygenxml.com/software_archive_editor.html

Interface



- Menu **File** pour créer des fichiers XML, des schémas (DTD, RNG, XSD) ou des transformations XSLT.
- Menu **Document** pour vérifier qu'un document est bien formé, le valider, le transformer.
- Icône  pour valider ou vérifier que le document est bien formé.
- Icône  pour exécuter les transformations.
- Zone de saisie et test des XPath :

XPath 2.0 SA 

Complément

Fichiers XML

Schémas XML

Transformations XSLT

4. Manipulation XML avec XPath

a) L'arbre du document XML

Il est possible de représenter un document XML sous forme d'arbre, tel que :

- L'arbre possède une racine / qui a comme fils l'élément racine
- l'élément racine est l'élément du document XML qui contient tous les autres

- chaque nœud a comme fils les éléments et le texte qu'il contient, ainsi que ses attributs.

Exemple : Fichier XML

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <document modele="ULCoursGeneral" code="BP-Incendie3_S1_E2_UL1">
3      <entete>
4          <identification>
5              <titre>L'assurance de la responsabilité de voisinage</titre>
6              <date>21/02/01</date>
7              <auteur>AEA</auteur>
8              <version>1.00</version>
9          </identification>
10     </entete>
11     <corps>
12         <contenu>
13             <paragraphe>Cette garantie est appelée : recours des voisins et des
14             tiers.</paragraphe>
15             <remarque>
16                 <paragraphe>L'image suivante <ressource URIsrc="img07.jpg"
17                 titre="Recours des voisins et des tiers" type="image"/>
18                 montre la
19                 garantie.</paragraphe>
20             </remarque>
21         </contenu>
22     </corps>
23 </document>

```

Exemple : Arbre XML

```

1  /
2  |document
3  |  |@modele = "ULCoursGeneral"
4  |  |@code = "BP-Incendie3_S1_E2_UL1"
5  |  |entete
6  |  |  |identification
7  |  |  |  |titre
8  |  |  |  |  |text() = "L'assurance de ..."
9  |  |  |  |date
10 |  |  |  |  |text() = "21/02/01"
11 |  |  |  |auteur
12 |  |  |  |  |text() = "AEA"
13 |  |  |  |version
14 |  |  |  |  |text() = "1.00"
15 |  |  |corps
16 |  |  |  |contenu
17 |  |  |  |  |paragraphe
18 |  |  |  |  |  |text() = "Cette garantie ..."
19 |  |  |  |  |remarque
20 |  |  |  |  |  |paragraphe
21 |  |  |  |  |  |  |text() = "L'image suivante"
22 |  |  |  |  |  |  |ressource
23 |  |  |  |  |  |  |  |@URIsrc = "img07.jpg"
24 |  |  |  |  |  |  |  |@titre = "Recours des voisins..."
25 |  |  |  |  |  |  |  |@type = "image"
26 |  |  |  |  |  |  |  |text() = "montre la garantie."

```

Remarque : Ordre des nœuds

L'ensemble des nœuds de l'arbre d'un document est muni d'un ordre, qui est celui de l'ordre dans le document XML sérialisé.

b) Introduction à XPath

Définition : Expression XPath

XPath est un langage d'expressions permettant de pointer sur n'importe quel élément d'un arbre XML depuis n'importe quel autre élément de l'arbre.

- Une expression XPath peut-être **absolue** (sa résolution est **indépendante** d'un contexte ou nœud courant : elle commence dans ce cas par `/`).
- Une expression XPath peut-être **relative** (sa résolution est **dépendante** d'un contexte ou nœud courant : elle ne commence dans ce cas pas par `/`, elle peut commencer par `./` (syntaxe développée)).

Fondamental

Une expression XPath renvoie

- **un *node-set*, ou ensemble de nœuds, c'est à dire un sous-arbre de l'arbre du document**
- **une chaîne de caractères**
- **un booléen**
- **un réel**

Exemple : Exemples d'expressions XPath

1	/document/entete/identification/titre
2	/document/@modele
3	corps//contenu
4	contenu/*
5	contenu/remarque[1]
6	../paragraphe
7	@type

Complément : Types de nœuds XPath

- root nodes
- element nodes
- text nodes
- attribute nodes
- namespace nodes
- processing instruction nodes
- comment nodes

<http://www.w3.org/TR/xpath/#data-model>³⁶

Complément

Pour une introduction à XPath : Brillant07 [Brillant07] pp.123-129

Complément : Voir aussi

L'arbre du document XML

Syntaxe XPath

c) Exercice

Soit le fichier XML ci-après.

Le nœud courant est un des éléments *terme*, écrivez 4 expressions XPath différentes permettant de renvoyer le titre du document :

1. Sachant que *titre* est unique dans tout le document.

36 - <http://www.w3.org/TR/xpath/#data-model>

2. Sachant que *titre* est le fils de l'élément racine *papier*.
3. Sachant que *titre* est le fils du père du père du nœud courant.
4. Sachant que *titre* est avant le nœud courant dans l'ordre du document.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <papier type="scientifique">
3      <titre>Réinterroger les structures documentaires</titre>
4      <sousTitre>De la numérisation à l'informatisation</sousTitre>
5      <auteur>Stéphane Crozat</auteur>
6      <auteur>Bruno Bachimont</auteur>
7      <resume>Nous proposons dans cet article d'aborder ...</resume>
8      <abstract>In this paper we define...</abstract>
9      <motsCles>
10         <terme>Ingénierie des connaissances</terme>
11         <terme>XML</terme>
12         <terme>Document</terme>
13     </motsCles>
14     <keywords>
15         <word>Knowledge engineering</word>
16         <word>XML</word>
17         <word>Document</word>
18     </keywords>
19     <publication date="2004-07-05"/>
20     <version maj="1" min="0"/>
21     <ressource
22         uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/sic_00001015.pdf"/>
    </papier>

```

1. //
2. /
3. ..
4. preceding

B. Exercice

1. Mon nom est personne

Modélisation XML

45 min

Question 1

Écrivez un document XML bien formé permettant de représenter des personnes, avec leur nom, leur prénom et leur âge.

On représentera deux personnes.

Indice :

Document bien formé

Balise

Exemple : Un document XML

Question 2

Écrivez un schéma au format DTD permettant de valider le document précédent, sachant que le nom et le prénom sont obligatoires, mais pas l'âge.

Indice :

Notion de document valide

Document Type Definition

*DTD : Déclaration d'éléments***Question 3**

Écrivez à nouveau le schéma, mais au formalisme RelaxNG, en utilisant le **typage des données** pour représenter l'age comme un *integer*.

Indice :

Relax NG : Syntaxe XML

RelaxNG : Types de données

Question 4

Écrivez des expressions XPath permettant de sélectionner :

- les noms des personnes, c'est à dire les éléments `nom` qui appartiennent à des éléments `personne`, qui appartiennent à l'élément `personnes` qui est à la racine.
- le nom de la seconde personne
- les noms des personnes dont l'age est renseigné
- les noms des personnes qui ont plus de 30 ans

Indice :

Introduction à XPath

Question 5

Testez vos propositions avec un éditeur XML validant.

Indice :

Présentation de oXygen XML Editor

2. Glossaire I

Soit le fichier XML suivant permettant de représenter un glossaire.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <glossaire>
3   <definition id="xml">
4     <terme>XML</terme>
5     <explication>XML est un méta-langage.</explication>
6   </definition>
7   <definition id="sgml">
8     <terme>SGML</terme>
9     <explication>SGML est un méta-langage.</explication>
10    <voirAussi ref="xml"/>
11  </definition>
12 </glossaire>

```

Pour les cinq questions suivantes, le nœud courant est `glossaire`.

Question 1

Que renvoie l'expression XPath suivante : `./*`

Question 2

Que renvoie l'expression XPath suivante : `./explication`

Question 3

Que renvoie l'expression XPath suivante : `//terme/text()`

Question 4

Écrire le XPath permettant de renvoyer les nœuds `terme`, qui ont `definition` comme père.

Question 5

Écrire le XPath permettant de renvoyer les nœuds `voirAussi` qui ont la valeur `xml` pour leur attribut `ref`.

Pour les trois questions suivantes, le nœud courant est le premier nœud `definition`, celui dont l'attribut `id="xml"`.

Question 6

Écrire le XPath permettant de renvoyer les nœuds `voirAussi` qui ont la valeur `xml` pour leur attribut `ref`.

Question 7

Écrire le XPath permettant de renvoyer les nœuds `definition` qui contiennent un élément `voirAussi` qui a la valeur `xml` pour son attribut `ref`.

Question 8

Écrire le XPath permettant de renvoyer le texte des nœuds `terme` fils des `definition` qui contiennent un élément `voirAussi` qui a la valeur `xml` pour son attribut `ref`.

Pour les deux questions suivantes, le nœud courant est un nœud `definition` quelconque.

Question 9

Écrire le XPath permettant de tester si le nœud courant contient un élément `voirAussi` qui se référence lui même (qui a la même valeur pour `ref` que l'attribut `id`).

Question 10

Écrire le XPath permettant de renvoyer les nœuds suivants qui se référencent eux-mêmes.

C. Devoir

1. On l'appelle Trinita

Soit la table `Personne` (`#id:integer`, `nom:varchar`, `prenom:varchar`, `age:integer`) avec `nom NOT NULL`, `prenom NOT NULL`.

Question 1

Créez la base de données relationnelle correspondante. Instanciez-là avec deux enregistrements.

Soit le fichier XML suivant :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <personnes>
3   <personne>
4     <nom>Personne</nom>
5     <prenom>Terence</prenom>
6   </personne>
7   <personne>
8     <nom>Trinita</nom>
9     <prenom>Terence</prenom>
10    <age>40</age>
11  </personne>
12 </personnes>

```

Question 2

Créer la vue SQL permettant de préparer la création de ce format XML en générant pour chaque enregistrement une ligne `<personne>`.

Indices :

```
1 <personne><nom>Personne</nom><prenom>Terence</prenom></personne>
2
   <personne><nom>Trinita</nom><prenom>Terence</prenom><age>40</age></perso
   nne>
```

On utilisera l'opérateur `||` : `SELECT x || ' ' || y FROM t.`

Introduction aux bases de données XML avec Oracle XMLType



IX

A. Cours

1. Bases de données XML

a) XML comme format d'échange

Fondamental

XML est un excellent candidat à l'échange de données entre bases de données

XML versus CSV

- XML est plus explicite que CSV (balises)
- On peut contrôler les données (schéma)
- Il y a une correspondance possible entre schéma XML et schéma de base de données (*mapping*)
- XML permet d'exporter des données relationnel-objet (arborescence)
- XML est plus verbeux, mais en général ce n'est pas un problème en général (sauf optimisation des transferts)

XML versus JSON

XML est à présent en concurrence avec JSON qui présente des caractéristiques similaires, avec quelques avantages (par exemple, il est plus compact et plus facilement *parsable*) et quelques défauts (il n'y a pas encore vraiment de langage de schéma associé).

Complément

Un *Internet Draft* de l'organisme de standardisation l'IETF (Internet Engineering Task Force) spécifie le langage *JSON Schema* qui est un formalisme de schéma pour JSON en JSON (sur le modèle de *XML Schema* pour XML).

<http://json-schema.org/>³⁷

37 - <http://json-schema.org/>

b) Stocker du XML en BD

SGBDR-XML

1. Mapping XML-Relationnel (en 1NF)
 - Chaque élément XML est extrait et stocké dans un attribut
 - Fonctionne pour les schémas simples, engendre des schémas très complexes et très fragmentés sinon, et la chute des performances
2. En BLOB
 - Les contenus XML sont stockés en BLOB (ou CLOB)
 - Des attributs dérivés de description peuvent être automatiquement extraits
 - Les BD n'est plus en 1NF, mais des fonctions de *parsing* fournies par le SGBD XML peuvent redonner accès à l'intérieur du contenu XML
 - Redondance entre les attributs dérivés et le BLOB, mais les attributs peuvent être recalculés à chaque mise à jour du contenu XML (ou implémentés comme des méthodes en RO)

Bases XML natives

- Basées sur un modèle logique de document XML et non plus un niveau logique relationnel
- Stockage physique spécifique (fichier XML, DOM persistant, ...)
- Gestionnaire offrant une logique transactionnelle orientée document XML
- Langage d'interrogation propre : XPath, XSL-XSLT, XQuery
- Plutôt pour les documents (volumineux) que les données

Exemple : eXists

Base de données native XML *Open Source*

<http://exist.sourceforge.net/>³⁸

38 - <http://exist.sourceforge.net/>

Exemple : Requête XQuery

```

-<mondial>
-<country car_code="AL" area="28750" capital="cty-cid-cia-Albania-Tirane" memberships="org-BSEC
org-CE org-CCC org-ECE org-EBRD org-FAO org-IAEA org-IBRD org-ICAO org-Interpol org-IDA org-IFRCS
org-IFC org-IFAD org-ILO org-IMO org-IMF org-IOC org-IOM org-ISO org-ICRM org-ITU org-Intelsat org-IDB
org-ANC org-OSCE org-OIC org-PFP org-UN org-UNESCO org-UNIDO org-UNOMIG org-UPU org-WFTU
org-WHO org-WIPO org-WMO org-WToO org-WTrO">
  <name>Albania</name>
  <population>3249136</population>
  <population_growth>1.34</population_growth>
  <infant_mortality>49.2</infant_mortality>
  <gdp_total>4100</gdp_total>
  <gdp_agri>55</gdp_agri>
  <inflation>16</inflation>
  <indep_date>1912-11-28</indep_date>
  <government>emerging democracy</government>
  <encompassed continent="europe" percentage="100"/>
  <ethnicgroups percentage="3">Greeks</ethnicgroups>
  <ethnicgroups percentage="95">Albanian</ethnicgroups>
  <religions percentage="70">Muslim</religions>
  <religions percentage="10">Roman Catholic</religions>
  <religions percentage="20">Christian Orthodox</religions>
  <border country="GR" length="282"/>
  <border country="MK" length="151"/>
  <border country="MNE" length="172"/>
  <border country="KOS" length="112"/>
-<city id="cty-cid-cia-Albania-Tirane" is_country_cap="yes" country="AL">
  <name>Tirane</name>
  <longitude>19.8</longitude>
  <latitude>41.3</latitude>
  <population year="87">192000</population>
</city>
-<city id="stadt-Shkoder-AL-AL" country="AL">
  <name>Shkoder</name>
  <longitude>19.2</longitude>
  <latitude>42.2</latitude>
  <population year="87">62000</population>
  <located_at watertype="lake" lake="lake-Skutarisee"/>
</city>
-<city id="stadt-Durres-AL-AL" country="AL">
  <name>Durres</name>
  <longitude>19.3</longitude>
  <latitude>41.3</latitude>
  <population year="87">60000</population>

```

The MONDIAL Database - <http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial.xml>

```

1 let $country := /mondial/country[name = 'France']
2 for $province in $country/province
3 order by $province/name
4 return $province/name

```

File

Slots

[1] --- empty ---
[2] --- empty ---
[3] --- empty ---
[4] --- empty ---
[5] --- empty ---
[6] --- empty ---
[7] --- empty ---
[8] --- empty ---
[9] --- empty ---
[10] --- empty ---

Paste example: Find spanish provinces and the Maximize

```
let $country := /mondial/country[name = 'France']
for $province in $country/province
order by $province/name
return $province/name
```

Send Clear Check Display: 50

Found 22 in 0 seconds. Showing items 1 to 22

- 1 <name>Alsace</name>
- 2 <name>Aquitaine</name>
- 3 <name>Auvergne</name>
- 4 <name>Basse Normandie</name>
- 5 <name>Bourgogne</name>
- 6 <name>Bretagne</name>
- 7 <name>Centre</name>
- 8 <name>Champagne Ardenne</name>
- 9 <name>Corse</name>
- 10 <name>Franche Comte</name>
- 11 <name>Haute Normandie</name>
- 12 <name>Ile de France</name>
- 13 <name>Languedoc Roussillon</name>
- 14 <name>Limousin</name>
- 15 <name>Lorraine</name>
- 16 <name>Midi Pyrenees</name>
- 17 <name>Nord Pas de Calais</name>
- 18 <name>Pays de la Loire</name>
- 19 <name>Picardie</name>
- 20 <name>Poitou Charentes</name>
- 21 <name>Provence Cote dAzur</name>
- 22 <name>Rhône Alpes</name>

<http://demo.exist-db.org/exist/sandbox/>

Complément

<http://peccatte.karefil.com/software/RBouret/xmlIBD.htm>³⁹

39 - <http://peccatte.karefil.com/software/RBouret/xmlIBD.htm>

c) Oracle XMLType : Relationnel-XML

Définition : XMLType

Oracle propose un type d'attribut permettant de stocker des documents XML : XMLType.

Par défaut, une colonne XMLType peut contenir n'importe quel document XML bien formé. De plus, le document peut être contraint par un schéma (*W3C XML Schema*).

- La fonction `EXTRACT('XPath')` permet de sélectionner un sous ensemble du document XML.
- La fonction `GETSTRINGVAL()` permet de récupérer le résultat sous la forme XML.

Syntaxe : LDD

```
1 CREATE TABLE tab (
2 ...
3 document XMLTYPE
4 ...
5 )
```

Syntaxe : Insertion

```
1 INSERT INTO tab (document) VALUES (XMLType('
2 <element>
3 ...
4 </element>
5 '))
```

Syntaxe : Sélection

```
1 SELECT t.document,EXTRACT('XPath').GETSTRINGVAL() FROM tab t;
```

d) Oracle XMLType : Table XML

Table XML

Il est possible de créer des tables XML, lorsque leur seule fonction est de stocker du XML

```
1 CREATE TABLE tab OF XMLTYPE
```

```
1 INSERT INTO tab VALUES (XMLType('
2 <element>
3 ...
4 </element>
5 '))
```

```
1 SELECT t.OBJECT_VALUE.EXTRACT('XPath').GETSTRINGVAL() FROM tab t
```

B. Exercices

1. On continue à l'appeler Trinita

Oracle XMLType (relationnel-XML)

Soit les fichiers XML suivants représentant les personnages des films "Trinita" (1967) et "Trinita est de retour" (1968).

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <personnes>
3   <personne>
4     <nom>Personne</nom>
5     <prenom>Terence</prenom>
6   </personne>
7   <personne>
8     <nom>Trinita</nom>
9     <prenom>Terence</prenom>
10    <age>40</age>
11  </personne>
12 </personnes>

```

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <personnes>
3   <personne>
4     <nom>Fonda</nom>
5     <prenom>Jack</prenom>
6     <age>53</age>
7   </personne>
8 </personnes>

```

Soit la table suivante en relationnel-XML : film (#nom:varchar(255), annee:number, pers:XMLType) .

Question 1

Créer une table Oracle correspondant en utilisant XMLType.

Indice :

Oracle XMLType : Relationnel-XML

Question 2

Insérer les données de l'exemple dans la table.

Question 3

Écrivez les requêtes SQL, en utilisant XPath, permettant de sélectionner :

- tous les noms de tous les personnages
- le nom du premier personnage de chaque film
- les noms des personnages dont l'age est renseigné
- les noms des personnages qui ont plus de 50 ans

Indice :

```
SELECT f.pers.EXTRACT('XPath').GETSTRINGVAL() AS ... FROM film f;
```

2. T'as le bonjour de Trinita

Oracle XMLType (table XML)

Soit les fichiers XML suivants représentant des films.

```

1 <film>
2   <nom>Trinita</nom>
3   <annee>1967</annee>
4   <personnes>
5     <personne>
6       <nom>Personne</nom>
7       <prenom>Terence</prenom>
8     </personne>
9     <personne>
10      <nom>Trinita</nom>
11      <prenom>Terence</prenom>
12      <age>40</age>
13    </personne>
14  </personnes>
15 </film>

```

```

1 <film>
2   <nom>Trinita est de retour</nom>
3   <annee>1968</annee>
4   <personnes>
5     <personne>
6       <nom>Fonda</nom>
7       <prenom>Jack</prenom>
8       <age>53</age>
9     </personne>
10  </personnes>
11 </film>

```

Question 1

Créer une table XML sous Oracle pour accueillir ces fichiers XML.

Indice :

Oracle XMLType : Relationnel-XML

Question 2

Insérer les données de l'exemple dans la table.

Question 3

Écrivez les requêtes SQL, en utilisant XPath, permettant de sélectionner :

- tous les noms de tous les personnages
- le nom du premier personnage de chaque film
- les noms des personnages dont l'age est renseigné
- les noms des personnages qui ont plus de 50 ans

Indice :

```
SELECT f.pers.EXTRACT('XPath').GETSTRINGVAL() AS ... FROM film f;
```

C. Devoirs

1. Documents

Soit une base de données de documents et d'auteurs, sachant que chaque auteur écrit plusieurs documents et chaque document est écrit par plusieurs auteurs.

Un document est composé de chapitres, eux-mêmes composés de sections, elles même-composées récursivement de sections ou de paragraphes.

Question 1

Proposer un diagramme UML du problème posé.

Question 2

Représenter ce modèle en utilisant le modèle relationnel-XML sous Oracle.

Question 3

Insérer des données de test.

Introduction aux data warehouses



A. Cours

1. Data warehouses et bases de données décisionnelles

Objectifs

Comprendre ce qu'est et à quoi sert un *data warehouse*.
Comprendre les différences entre un *data warehouse* et une base de données transactionnelle.

a) Décisionnel

Définition

« Le système d'information décisionnel est un ensemble de données organisées de façon spécifiques, facilement accessibles et appropriées à la prise de décision [...].
La finalité d'un système décisionnel est le pilotage d'entreprise.
Les systèmes de gestion sont dédiés **aux métiers** de l'entreprise [...].
Les systèmes décisionnels sont dédiés **au management** de l'entreprise [...].
(Goglin, 2001, pp21-22) »



Synonymes : informatique décisionnelle, *business intelligence*, BI★

b) Data warehousing

Définition : Définition historique de Inmon

« A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions. The data warehouse contains granular corporate data.
(Inmon, 2002, p31) »



Définition

Un data warehouse (DW★) est une base de données construite par copie et réorganisation de multiples sources (dont principalement le système transactionnel de l'entreprise), afin de servir de source de données à des applications décisionnelles :

- il agrège de nombreuses données de l'entreprise (**intégration**) ;
- il mémorise les données dans le temps (**historisation**) ;
- il les organise pour faciliter les requêtes de prise de décision (**optimisation**).

(Goglin, 2001, p27) [(Goglin, 2001)]

Synonymes : entrepôt de données, base de données décisionnelle

Fondamental

L'objectif du data warehouse est de permettre des requêtes sur de grands ensembles des données, la plupart du temps sous forme d'agrégats (GROUP BY) afin d'en obtenir une vision synthétique (propre à la prise de décision).

Remarque

Le data warehouse dédié au décisionnel est séparé du système transactionnel dédié à la gestion quotidienne.

Complément : Voir aussi

Data warehouse et data mart

c) Différence entre un DW et un système transactionnel

BD transactionnelle

Une base données classique est destinée à assumer des **transactions** en temps réel :

- Ajout, mise à jour suppression de données
- Questions sur des données identifiées ou questions statistiques

Datawarehouse

Un DW★ est uniquement destiné à l'exécution de **questions statistiques** sur des données statiques (ou faiblement dynamiques).

PRIMITIVE DATA/OPERATIONAL DATA

- application oriented
- detailed
- accurate, as of the moment of access
- serves the clerical community
- can be updated
- run repetitively
- requirements for processing understood a priori
- compatible with the SDLC
- performance sensitive
- accessed a unit at a time
- transaction driven
- control of update a major concern in terms of ownership
- high availability
- managed in its entirety
- nonredundancy
- static structure; variable contents
- small amount of data used in a process
- supports day-to-day operations
- high probability of access

DERIVED DATA/DSS DATA

- subject oriented
- summarized, otherwise refined
- represents values over time, snapshots
- serves the managerial community
- is not updated
- run heuristically
- requirements for processing not understood a priori
- completely different life cycle
- performance relaxed
- accessed a set at a time
- analysis driven
- control of update no issue
- relaxed availability
- managed by subsets
- redundancy is a fact of life
- flexible structure
- large amount of data used in a process
- supports managerial needs
- low, modest probability of access

Un changement d'approche, extrait de (Inmon, 2002, p15)

d) Implémentation du DW avec un SGBDR

Fondamental

Les deux problématiques fondamentales des DW★ sont l'**optimisation** et la **simplification** : comment rester **performant** et **lisible** avec de très gros volumes de données et des requêtes portant sur de nombreuses tables (impliquant beaucoup de jointures) ?

On utilise massivement :

- **Les vues concrètes** : Un data warehouse procède par copie depuis le ou les systèmes transactionnels
- **La dénormalisation** : Un data warehouse est hautement redondant

Fondamental

Le caractère **statique** du data warehouse efface les inconvénients de ces techniques lorsqu'elles sont mobilisées dans des systèmes transactionnels.

Rappel

Dénormalisation

Vues concrètes

e) Modélisation logique de données en étoile

Définition : Le modèle en étoile

Le **modèle en étoile** est une représentation fortement **dénormalisée** qui assure un haut niveau de performance des requêtes même sur de gros volumes de données.

Exemple

Exemple de modèle dimensionnel en étoile

Complément : Modèle en flocon

Le modèle en flocon est aussi un modèle dénormalisé, mais un peu moins que le modèle en étoile : il conserve un certain niveau de décomposition pour chaque dimension prise isolément.

Complément : Voir aussi

Modélisation en flocon

f) Extraction Transformation Loading

Définition : ETL

L'ETL (*Extraction Transformation Loading*) est le processus de copie des données depuis les tables des systèmes transactionnels vers les tables du modèle en étoile du data warehouse.

Exemple

Exemple de modèle dimensionnel en étoile

Remarque

Les tables du modèle dimensionnel peuvent être vues comme des **vues concrètes** sur le système transactionnel, à la nuance que des transformations (correction d'erreur, extrapolation...) peuvent avoir été apportées dans le processus ETL.

2. Pour aller plus loin

Complément

<https://stph.scenari-community.org/dwh/>⁴⁰

B. Exercice

1. Étude de cas : Fantastic

Cet exercice constitue une simulation de modélisation dimensionnelle basée sur une étude de cas.

a) Approche générale de modélisation

Rappel

La modélisation en étoile

Fondamental

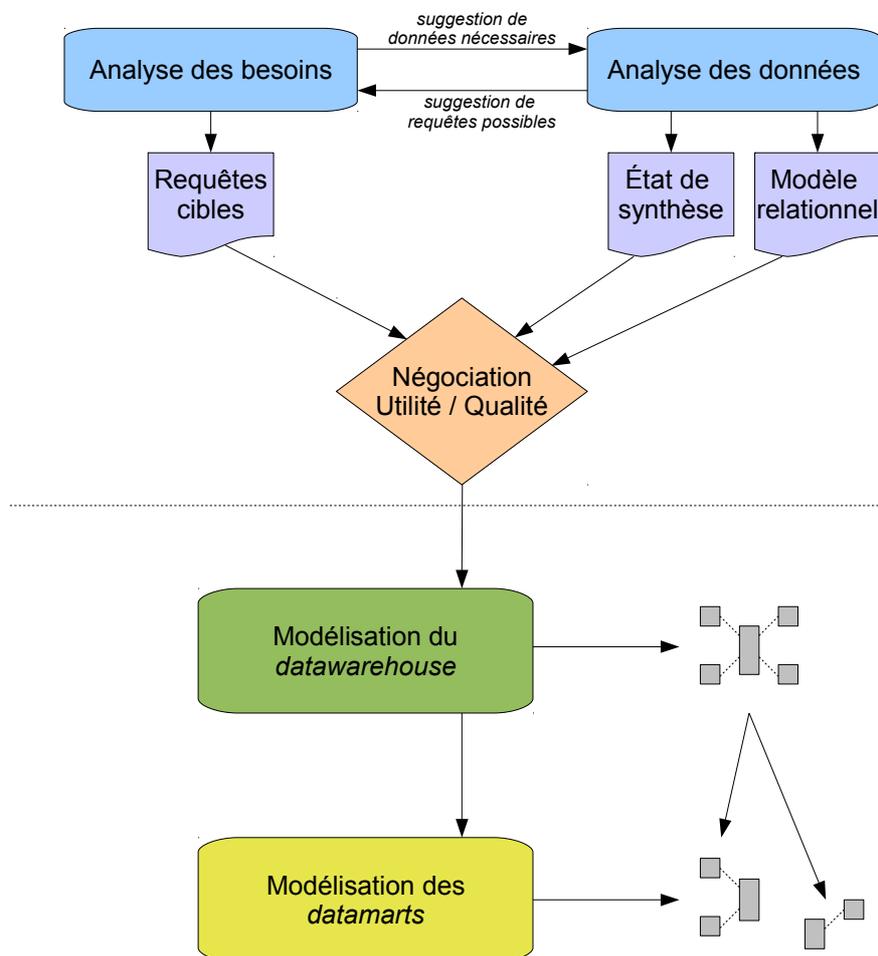
Un modèle dimensionnel est le résultat :

- **d'une analyse des besoins : ce que je souhaite étudier.**
- **d'une analyse des données disponibles : ce que je peux étudier.**

40 - <https://stph.scenari-community.org/dwh/>

Méthode : Méthode générale de modélisation

1. Analyse des données
 - a. Étude des sources de données (quantification, analyses générales)
 - b. Qualification des données (qualité et intérêt)
 - c. Intégration logique des données (simulation d'un schéma relationnel virtuel)
 - d. Normalisation du schéma virtuel en 3NF pour en avoir une vue cohérente
2. Analyse des besoins clients
 - a. Exprimer les besoins sous la forme de requêtes décisionnelles
 - b. Réaliser les vues hiérarchiques pour chaque requête
3. Sélectionner les requêtes qui seront effectivement réalisables en fonction des données disponibles
4. Conception du data warehouse et des data marts
 - a. Séparer les requêtes en fonction de la granularité de la table des faits (grain fin des ventes, grain plus grossier du ticket de caisse, etc.)
 - b. Créer un data warehouse intégrant toutes les requêtes de grain fin
 - c. Extraire un data mart par niveau de grain supérieur et/ou pour des thématiques particulières nécessitant par exemple une pré-agrégation



Graphique 1 Éléments méthodologiques pour la modélisation dimensionnelle

Remarque

Il est généralement intéressant de paralléliser les tâches d'analyse des besoins et d'analyse des données.

En particulier il est inutile d'aller trop loin dans l'expression de besoins que l'on sait **a priori** impossibles à satisfaire pour cause d'absence de donnée ou d'absence de donnée exploitable.

Rappel : Informations

Il est conseillé de conserver certains champs d'information dans le modèle dimensionnel, même s'ils ne seront pas exploités pour les calculs ou les agrégats.

Cela permettra par exemple d'identifier des enregistrements, comme les désignations de produits.

On pourra noter en italique ces champs dans le modèle dimensionnel.

b) Présentation du cas

i Cas Fantastic : Problème posé

Vous travaillez en tant qu'ingénieur spécialisé dans les systèmes décisionnels au siège de l'entreprise française "Fantastic".

L'entreprise "Fantastic" vend principalement des ouvrages de divertissement de type science fiction, thriller, policier... Elle dispose pour cela de plusieurs magasins de vente dans les centres des grandes villes en France.

La direction de l'entreprise souhaite faire une étude large sur les ventes de l'année passée afin de prendre des orientations stratégiques nouvelles : ouverture de nouveaux magasins, fermeture ou transfert de magasins mal implantés, extension territoriale à de nouveaux départements français, réorganisation des directions, réorientation du marketing, élargissement ou réduction du catalogue, etc.

Fondamental

La question posée est donc : quels sont les facteurs sur lesquels l'on pourrait jouer pour augmenter les ventes ?

Elle vous charge dans ce cadre de mettre en place une solution logicielle permettant d'intégrer les données pertinentes et de pouvoir les interroger efficacement sous des angles divers.

Notons que bien entendu, la direction éclairée de l'entreprise ne compte pas se fier à ces seuls facteurs de ventes pour prendre ses décisions, mais bien privilégier les facteurs sociaux et territoriaux, en dialoguant avec ses salariés et ses clients, pour maintenir sa mission culturelle et son rôle d'entreprise citoyenne. Votre posture d'ingénieur est bien entendu de se préoccuper de ces dimensions fondamentales, même si elles seront largement ignorées dans le cadre de cet exercice à vocation essentiellement technique. Elle pourront néanmoins être brièvement abordées en marge de vos rapports d'analyse.

ii Cas Fantastic : Données disponibles

Attention : Installation

Le cas repose sur une installation spécifique, nous utiliserons les termes génériques suivantes :

- ***oracle.univ.fr*** le nom d'un serveur Linux hébergeant une instance Oracle
- ***prof*** le nom d'un schéma Oracle géré par l'enseignant
- ***etuXXX*** le nom d'une série de schémas Oracle disponibles pour les étudiants
- ***profdir*** le nom d'un compte Linux géré par l'enseignant sur *oracle.univ.fr*
- ***profdir*** le chemin du compte de *prof* sur le serveur Linux
- ***etuXXX*** le nom d'une série de comptes Linux disponibles pour les étudiants
- ***www.univ.fr/fantastic*** le nom d'un site Web géré par l'enseignant

Exemple : Installation UTC

En 2016, pour le cours AI07 à l'UTC l'installation est la suivante :

- *sme-oracle.sme.utc* (serveur Oracle)
- *ai07* (schéma Oracle et compte Linux enseignant)

- `/volsme/users/ai07` (chemin du compte Linux enseignant)
- `ai07aXXX` (schémas Oracle et comptes Linux étudiants)
- <https://stph.scenari-community.org/ai07/fantastic>⁴¹ (site Web de mise à disposition des données)

Catalogue des livres

Une base Oracle contient le catalogue complet de l'entreprise que chaque magasin a à sa disposition.

- Cette base, composée d'une seule table publique `catalogue`, est disponible sur le serveur Oracle sous le schéma `prof`.

Fichier des ventes

Un fichier *Fantastic* contient une consolidation de l'ensemble des ventes de l'année passée réalisées dans chaque magasin.

- Ces données sont disponibles sous la forme d'un fichier CSV dans un répertoire du serveur `oracle.univ.fr` : `profdir/data`
- La structure du fichier est : Numéro de ticket, date de ticket, produit, magasin (comme c'est rappelé dans la documentation HTML associée)

Fichier des magasins

Un fichier ODS géré par la direction marketing contient pour chaque magasin l'organisation des rayonnages : `marketing.ods`

- Le responsable des ventes de chaque département décide de l'organisation des rayonnages des magasins de son département.
- Il existe 3 types de rayonnage : par Auteur (A), par Année (Y), par Éditeur (E)
- Le fichier est déposé sur www.univ.fr/fantastic/data1

Données géographique sur les départements

Un stagiaire a trouvé sur Internet un fichier permettant de connaître la population de chaque département, présageant que cette information sera utile.

- Le stagiaire parvient à trouver une information un peu datée qui pourra suffire sous la forme d'un fichier CSV : `departementsInsee2003.txt`.
- Le fichier ainsi que sa documentation en HTML est déposé sur www.univ.fr/fantastic/data1

Méthode

Inspecter les données pour chaque source :

1. Se connecter à la base Oracle avec SQL Developer et inspecter le schéma de la table Oracle.
2. Ouvrir un terminal et se connecter en `ssh` au serveur.
Utiliser la commande Unix `more` pour regarder les premières lignes du fichier *Fantastic*.
3. Récupérer le fichier CSV `departementsInsee2003.txt` et l'ouvrir avec un éditeur de texte.
4. Récupérer le fichier ODS et l'ouvrir avec un traitement de texte.

iii Exemple d'inspection des données

1 Fichier "Fantastic"

Documentation

Relevé des ventes de livres Fantastic pour l'année 2015

41 - <https://stph.scenari-community.org/ai07/fantastic>

1. Ticket number
2. Date
3. ISBN
4. Store

Commande 1

```
1 head -10 Fantastic
```

```
1 "142282000001";2015-10-10;"769";"M143"
2 "0";;"9782841724680";"M119"
3 "082229000003";2015-08-18;"9782266186315";"83"
4 "082229000003";2015-08-18;"9782266186315";"M83"
5 "082229000003";2015-08-18;"9782266186315";"M83"
6 "028093000004";2015-04-04;"9780765341600";"M29"
7 "115072000005";2015-03-14;"9782290348789";"M116"
8 "040187000006";2015-07-07;"9782702141045";"M41"
9 "031002000007";2015-01-03;"552";"M32"
10 "055114000008";2015-04-25;"9782207301999";"M56"
```

Commande 2

```
1 wc -l Fantastic
```

```
1 512018
```

2 Fichier "departementsInsee2003.txt"

Documentation

Liste des départements français, 2003

1. **Department** Départements français métropolitains
2. **DptName** Nom du département
3. **Population** Population du département en 2003

Commande 1

```
1 head -10 departementsInsee2003.txt
```

```
1 "01";"Ain";"529378"
2 "02";"Aisne";"552320"
3 "03";"Allier";"357110"
4 "04";"Alpes-de-Haute-Provence";"144809"
5 "05";"Hautes-Alpes";"126636"
6 "06";"Alpes-Maritimes";"1022710"
7 "07";"Ardèche";"294522"
8 "08";"Ardennes";"299166"
9 "09";"Ariège";"142834"
10 "10";"Aube";"301388"
```

Commande 2

```
1 wc -l departementsInsee2003.txt
```

```
1 95
```

3 Fichier "marketing.ods"

Documentation

Rayonnage : Le responsable des ventes de chaque département décide de l'organisation des rayonnages des magasins de son département. Il existe 3 types de rayonnage : par Auteur (A), par Année (Y), par Éditeur (E)

Ouverture du fichier dans LibreOffice (extrait)

Magasin	Département	Rayonnage		Rayon Bestseller
M62	1	E	Editor	1
M101	2	A	Author	1
M122	4	A	Author	0
M130	6	A	Author	1
M139	6	A	Author	1
M93	6	A	Author	1
M119	8	A	Author	0
M69	8	A	Author	0
M13	10	E	Editor	0
M10	13	A	Author	1

(153 lignes, dont l'entête)

4 Table "catalogue"

Commande 1

```
1 DESCRIBE catalogue
```

1	Name	Null	Type
2	-----	-----	-----
3	REF	NOT NULL	NUMBER (38)
4	ISBN	NOT NULL	VARCHAR2 (13)
5	TITLE	NOT NULL	VARCHAR2 (255)
6	AUTHORS	NOT NULL	VARCHAR2 (255)
7	LANGUAGE		VARCHAR2 (3)
8	PUBDATE		VARCHAR2 (25)
9	PUBLISHER		VARCHAR2 (255)
10	TAGS		VARCHAR2 (255)
11	GENRE		VARCHAR2 (255)

Commande 2

```
1 SELECT * FROM catalogue
2 WHERE ROWNUM < 11;
```

REF	ISBN	TITLE	AUTHORS	LANGUAGE	PUBDATE	PUBLISHER	TAGS	GENRE
2778721	[Anita Blake-1]Narcisse Enchaîné	Hamilton,Laurell Kaye	fra	2001-01-01T23:00:00+00:00 ?	Fantastique, Roman	Fantastique	Fantastic	
2833779	[La Communauté du Sud-1] Quand le danger rôde	Harris,Charlaine	fra	2001-01-01T23:00:00+00:00 ?	Fantastique	Fantastique	Fantastic	
315141	[Assassin Royal-07] Le prophète blanc	Hobb,Robin	fra	2001-01-01T21:00:00+00:00 ?	Fantasy	Fantasy	Fantastic	
2883828	[Assassin Royal-08] La secte maudite	Hobb,Robin	fra	2001-01-01T21:00:00+00:00 ?	Fantasy	Fantasy	Fantastic	
2770712	[Merry Gentry-1] Le baiser des ombres	Hamilton,Laurell Kaye	fra	2000-01-14T23:00:00+00:00 ?	Fantastique	Fantastique	Fantastic	
339233	[Anita Blake-9]Papillon d'Obsidienne	Hamilton,Laurell Kaye	fra	2000-01-01T23:00:00+00:00 ?	Fantastique, Roman	Fantastique	Fantastic	
2882827	[Les Aventuriers de la mer-07] Le seigneur des Trois Règnes	Hobb,Robin	fra	2000-01-01T21:00:00+00:00 ?	Fantasy	Fantasy	Fantastic	
317143	[Les Aventuriers de la mer-08] Ombres et flammes	Hobb,Robin	fra	2000-01-01T21:00:00+00:00 ?	Fantasy	Fantasy	Fantastic	
2904848	[Les Aventuriers de la mer-09] Les marches du trône	Hobb,Robin	fra	2000-01-01T21:00:00+00:00 ?	Fantasy	Fantasy	Fantastic	
259214	[Robert Langdon-1]Ange et démons	Brown,Dan	fra	1999-12-31T21:00:00+00:00 ?	Policier, Thriller Scientifique, BROWN, Langdon	Crimel	Fantastic	

Commande 3

```
1 SELECT COUNT(*) FROM catalogue;
```

c) Étude des données

Objectifs

Savoir faire une étude des données existantes

i Étude séparée des sources données

Méthode

Pour chaque source de données identifiée on proposera une synthèse avec les informations suivantes :

- Nom, origine précise, nombre de lignes de la source
- Nom, type et description des colonnes
- Qualité
 - 0 : données inexploitable
 - 1 : données peu exploitables (traitements incertains)
 - 2 : données exploitables après traitements
 - 3 : données exploitables sans traitement
- Utilité
 - 0 : données sans intérêt
 - 1 : données utiles pour la documentation uniquement
 - 2 : données utiles a priori
 - 3 : données utiles avec certitude
- Commentaires.

Exemple

Nom	Description	Type	Qualité	Utilité	Commentaire
isbn	Identifiant international d'un livre publié	char(3) et char(13)	3	3	Certaines ISBN ne sont pas corrects (3 caractères au lieu de 13) ; clé de référencement dans le fichier de ventes
titre	Titre du livre	varchar(255)	2	1	
auteur	Auteur(s) du livre	varchar(255)	1	2	Peut contenir plusieurs auteurs ; certains auteurs sont inscrits différemment d'un livre à l'autre
...					

Tableau 5 Table Oracle oracle.utc.fr/schema.table [1000 lignes]

ii Étude intégrée des sources de données

Méthode

Afin d'avoir une vision globale de l'ensemble des données, il est conseillé de rétro-concevoir :

- une représentation relationnelle des données telles qu'elles existent
- une représentation relationnelle des données idéalisées (telles qu'elles existeraient si elles étaient normalisées dans une même BD)
- une représentation conceptuelle en UML

d) Cas Fantastic : Étude des données

[1h]

Afin de réaliser votre travail, l'entreprise vous met à disposition les données suivantes.

Dans le contexte de cet exercice, les données vous sont livrées *a priori*, notez que dans un contexte réel, vous aurez la plupart du temps à rechercher vous même les données qui peuvent exister.

Données disponibles

Question 1

Établissez le modèle **relationnel** sous-jacent aux données présentes.

Indice :

Pour initier une connexion ssh : `ssh user@serveur`

Question 2

Étudiez les données dont vous disposez et proposez une synthèse des données disponibles pour chaque source.

Indice :

Pour compter les lignes d'un fichier texte sous Linux, on peut utiliser la commande `wc -l`

Question 3

Afin de clarifier les données et leur organisation, rétro-concevez un modèle relationnel **normalisé** en troisième forme normale unifiant toutes les données grâce à l'identification de clés primaires et l'expression de clé étrangères.

Question 4

Rétro-concevez le modèle **conceptuel** en UML correspondant au modèle relationnel normalisé (un modèle UML peut être plus facile à utiliser ensuite).

e) Étude des besoins utilisateurs

Objectifs

Savoir formaliser une étude de besoins sous forme de requêtes multidimensionnelles

i Requête décisionnelle

Définition : Requête décisionnelle

Une requête décisionnelle exprime toujours la mesure d'une quantification de **faits** par rapport à des **dimensions**, sous une forme du type : "Quelle a été la quantité de ... en fonction de ...".

Synonyme : Vue, requête multidimensionnelle

Fondamental

- **Les faits sont des grandeurs que l'on cherche à mesurer (prix, quantité...)**
- **Les dimensions sont des axes d'analyse (date, lieu, produit, personne...)**

Syntaxe

```
1 quantité de faits
2 / dimension1
3 / dimension2
4 ...
```

Exemple

"Quelle a été la quantité de produits vendus en fonction des départements et des mois de l'année."

```
1 quantité de produits vendus
2 / département
3 / mois
```

ii Rapport

Définition : Rapport

La réponse à une requête décisionnelle est un rapport, généralement sous une forme tabulaire ou graphique.

Synonyme : État

Exemple

01												02												03											
J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D
04												05												06											
J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D
07												08												09											
J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D

Exemple de rapport

Méthode

Les besoins des utilisateurs s'expriment en général plus facilement sous la forme d'exemples de rapports recherchés.

iii Hiérarchie

Définition : Hiérarchie

Une hiérarchie est un ensemble de paramètres d'étude de granularité croissante appartenant à une même dimension au sein d'un modèle décisionnel.

Exemple

1	quantité de produits vendus
2	/ lieu (département, région)
3	/ date (jour, mois, trimestre, semestre)

Remarque

Une même dimension peut contenir plusieurs hiérarchies.

f) Cas Fantastic : Étude des besoins

[30 min]

À partir de l'étude des besoins sommaire effectuée par un de vos collègues, et en fonction des données disponibles, exprimer les **requêtes cibles** de votre système.

Le contexte de l'exercice ne permet pas de dialoguer réellement avec des utilisateurs, **en situation réelle il faudra développer cette phase de recueil des besoins des utilisateurs**. Vous pourrez amendez l'indicateur d'utilité des données en fonction de cette étude.

Question 1

« La direction marketing est en charge de l'implantation des magasins dans les départements et de l'organisation des rayonnages (type de rangement et présence de rayons spécifiques pour les best-sellers). Elle cherche à savoir si l'organisation du rayonnage des magasins a une influence sur les volumes ventes, et si cela varie en fonction des jours de la semaine ou de certaines périodes de l'année. Elle voudrait également savoir si certains magasins ou départements sont plus dynamiques que d'autres. »

Question 2

« La direction éditoriale se demande si certains livres se vendent mieux à certaines dates et/ou dans certains magasins ou départements. Elle aimerait également savoir si certains auteurs ou éditeurs se vendent mieux, et s'il existe un lien entre l'ancienneté des livres et les ventes. Elle se demande aussi si certaines périodes sont plus propices que d'autres à l'écoulement des livres les plus anciens. »

g) Modélisation en étoile

i Table des faits

Définition

« A row in a fact table corresponds to a measurement. A measurement is a row in a fact table. All the measurements in a fact table must be the same grain. (Kimball, Ross, 2008, p.17) »

« Fact tables express the many-to-many relationships between dimensions in dimensional models. (Kimball, Ross, 2008, p.19) »

Remarque

La table des faits est (dans la plupart des cas) la table la plus volumineuse (avec le plus grand nombre de lignes) du modèle.

Exemple

Daily Sales Fact Table
Date Key (FK)
Product Key (FK)
Store Key (FK)
Quantity Sold
Dollar Sales Amount

Exemple de table des faits (Kimball, Ross, 2008, p.17)

Fondamental : Faits additifs et numériques

« The most useful facts are numeric and additive. (Kimball, Ross, 2008, p.17) »

Méthode : Granularité minimale

« Preferably you should develop dimensional models for the most atomic information captured by a business process. Atomic data is the most detailed information collected; such data cannot be subdivided further. (Kimball, Ross, 2008, p.34) »

Méthode : Granularité des data marts

Pour un data mart on peut pré-agrégé sur un grain plus gros que le data warehouse : des colonnes d'agrégation (somme, moyenne, compte...) peuvent alors apparaître pour rendre compte statistiquement d'informations perdues à l'agrégation.

ii Table des dimensions

Définition

« Dimension tables are the entry points into the fact table. [...] The dimension implement the user interface to the data warehouse. (Kimball, Ross, 2008, p.20) »

Exemple

Product Dimension Table
Product Key (PK)
Product Description
SKU Number (Natural Key)
Brand Description
Category Description
Department Description
Package Type Description
Package Size
Fat Content Description
Diet Type Description
Weight
Weight Units of Measure
Storage Type
Shelf Life Type
Shelf Width
Shelf Height
Shelf Depth
... and many more

Exemple de table de dimension (Kimball, Ross, 2008, p.20)

Conseil : Intelligibilité

« The best attributes are textual and discrete. Attributes should consist of real words rather than cryptic abbreviations. (Kimball, Ross, 2008, p.20) »

h) Cas Fantastic : Modélisation

À partir de l'étude des données et des besoins utilisateurs, proposez un modèle **dimensionnel** pour un **data warehouse** permettant de traiter la question générale qui a été posée.

Question 1

Proposez une modélisation dimensionnelle **en étoile** pour chaque contexte d'usage (directions marketing et éditoriale) :

1. identifiez la table des faits
2. identifiez les dimensions en intégrant les vues exprimées (utilisez le rapport qualité/utilité pour décider des données que vous souhaitez conserver)

Question 2

Intégrer vos deux sous-modèles pour proposer le modèle de votre **data warehouse**.

Vous pourrez augmenter vos dimensions de données disponibles bien que non identifiées a priori lors de l'analyse des besoins.

Question 3

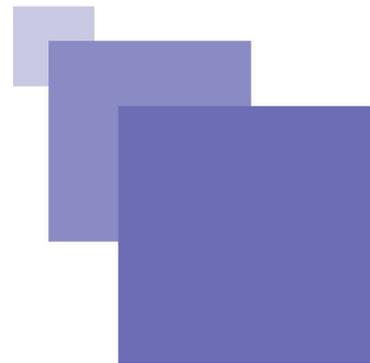
Établissez les métadonnées du modèle dimensionnel : décrivez chaque données (type précis, description...) ; identifiez la clé primaire de chaque dimension, ainsi que les informations descriptives.

Question 4

Proposez la représentation logique finale de votre modèle dimensionnel, en affichant :

- la table des faits et les dimensions
- l'ensemble des attributs
- les clés primaires et étrangères
- les types de données
- les hiérarchies

Glossaire



Constructeur d'objet

En programmation orientée objet, un constructeur d'objet est une méthode particulière d'une classe qui permet d'instancier un objet de cette classe. L'appel à cette méthode de classe a donc pour conséquence la création d'un nouvel objet de cette classe.

Impedance mismatch

Le terme d'*impedance mismatch* renvoie au décalage qui peut exister entre le niveau d'abstraction de deux langages qui ont à travailler sur des structures de données communes, par exemple un langage applicatif objet et un langage de données relationnel. L'*impedance mismatch* a des conséquences négatives en terme de complexification de l'implémentation et en terme de performance, puisqu'il faut constamment passer d'une structure de données à l'autre.

Sérialisation

Processus consistant à enregistrer des données en mémoire vive (par exemple des objets) sous une forme permettant leur persistance, typiquement sur une mémoire secondaire.

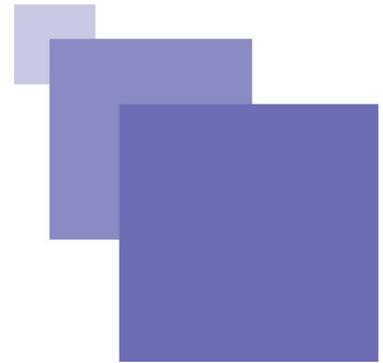
XSL-FO

XSL-FO (FO pour Formatting Objects) est la seconde partie du standard W3C « Extensible Stylesheet Language Family ». Il propose un langage XML de mise en forme de documents imprimables.

Exemple d'extrait de code FO :

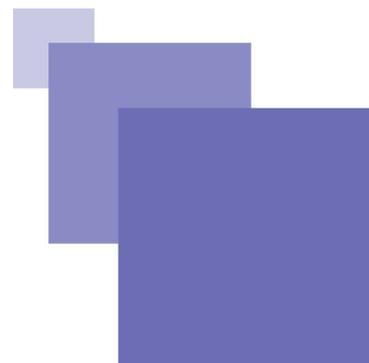
- `<fo:block font-family="Times" font-size="12pt">Ceci est un paragraphe en police Times de taille 12pt</fo:block>`.

Signification des abréviations



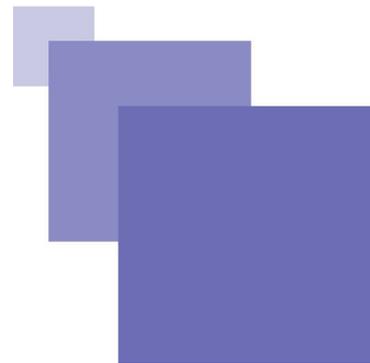
- 1NF	First Normal Form
- 3NF	Third Normal Form
- API	Application Program Interface
- BD	Base de Données
- BI	Business Intelligence
- CSV	Comma Separated Values
- DW	Data Warehouse
- E-A	Entité-Association
- IDE	Integrated Development Environment (Environnement de Développement Intégré)
- LDD	Langage de Définition de Données
- LMD	Langage de Manipulation de Données
- MCD	Modèle Conceptuel de Données
- MLD	Modèle Logique de Données
- OID	Object Identifier
- OS	Operating Système (Système d'Exploitation)
- POO	Programmation Orientée Objet
- SGBD	Système de Gestion de Bases de Données
- SGBDOO	Système de Gestion de Bases de Données Orientées Objets
- SGBDR	Système de Gestion de Bases de Données Relationnelles
- SGBDRO	Système de Gestion de Bases de Données Relationnelles-Objets
- SQL	Structured Query Language
- UML	Unified Modeling Language
- W3C	World Wide Web Consortium
- XML	eXtensible Markup Language

Bibliographie



- [[**Crozat, 2007**]] STÉPHANE CROZAT, *Scenari, la chaîne éditoriale libre*, Accès Libre, Eyrolles, 2007.
- [[**Dupoirier, 1995**]] GÉRARD DUPOIRIER, *Technologie de la GED : Techniques et management des documents électroniques*, Hermes, 1995.
- [[**Goglin, 2001**]] Goglin J.-F. (2001, 1998). *La construction du datawarehouse : du datamart au dataweb*. Hermes, 2ème édition.
- [[**Inmon, 2002**]] Inmon W.-H. (2002, 1990). *Building the data warehouse*. Wiley Publishing, 3rd edition.
- [[**Kimbal, Ross, 2003**]] Kimball R., Ross M. (2003). *Entrepôts de données : guide pratique de modélisation dimensionnelle*. Vuibert.
- [[**Kimball, Ross, 2008**]] Kimball R., Ross M. (2008, 2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley Publishing, second edition.
- [[**Kimball et al., 2008**]] Kimball R., Ross M., Thornthwaite W., Mundy J., Becker B. (2008, 1998). *The Data Warehouse Lifecycle Toolkit*. Wiley Publishing, second edition.
- [[**Abbey01**]] ABBEY MICHAEL, COREZ MICHAEL, ABRAMSON IAN, *Oracle 9i : Notions fondamentales*, CampusPress, 2001.
- [[**André89**]] JACQUES ANDRÉ, RICHARD FURUTA, VINCENT QUINT, *Structured documents*, Cambridge University Press, 1989.
- [[**Bachimont07**]] BRUNO BACHIMONT, *Ingénierie des connaissances et des contenus : le numérique entre ontologies et documents*, Lavoisier, Hermès, 2007
- [[**Brillant07**]] ALEXANDRE BRILLANT, *XML : Cours et exercices*, Eyrolles, 2007 [ISBN 978-2212126914]
- [[**Bruchez, 2015**]] BRUCHEZ RUDI. 2015. *Les bases de données NoSQL et le BigData : Comprendre et mettre en oeuvre*. 2ème édition, Eyrolles.
- [[**Espinasse, 2013**]] ESPINASSE BERNARD. 2013. *Introduction aux systèmes NoSQL (Not Only SQL)*. http://www.lsis.org/espinasseb/Supports/BD/BD_NOSQL-4p.pdf.
- [[**w3resource, 2015**]] W3RESOURCE. 2015. *NoSQL introduction*. <http://www.w3resource.com/mongodb/nosql.php>.

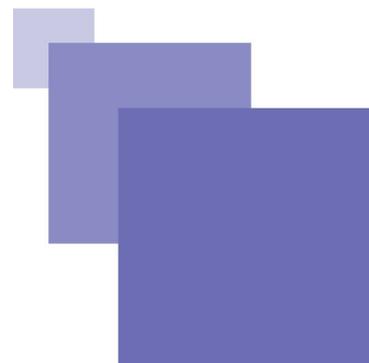
Webographie



[w_w3c.org/XML] XML, <http://www.w3.org/XML> .



Index



<i>1:1</i>	p.67	<i>DTD</i>	p.92	<i>Objet</i>	p.36, 54, 64, 65, 66, 73, 74
<i>1:N</i>	p.67	<i>Élément</i>	p.92	<i>OID</i>	p.65, 66, 74, 74, 75, 78
<i>1NF</i>	p.41	<i>Élément</i>	p.	<i>PL/SQL</i>	p.74
<i>Application</i>	p.	<i>Enregistrement</i>	p.51	<i>Procédure</i>	p.
<i>Association</i>	p.67, 67, 68	<i>Enregistrement imbriqué</i>	p.41,	<i>Redondance</i>	p.
<i>Attribut</i>	p.90, 92	43, 50		<i>REF</i>	p.75
<i>Attribut composite</i>	p.45	<i>Entité</i>	p.66	<i>Référence</i>	p.65, 66, 74
<i>Attribut dérivé</i>	p.45	<i>Etat</i>	p.	<i>Relationnel</i>	p.35, 35
<i>Attribut multi-valué</i>	p.45	<i>Fonction</i>	p.39	<i>Relationnel-objet</i>	p.37
<i>Balise</i>	p.	<i>FUNCTION</i>	p.39	<i>Schéma</i>	p.70, 79
<i>Bien formé</i>	p.85	<i>Héritage</i>	p.70, 79	<i>SELECT</i>	p.51, 75
<i>Classe</i>	p.38	<i>INSERT INTO</i>	p.51, 54, 74, 78	<i>SGBDOO</i>	p.36
<i>Clé étrangère</i>	p.55, 67	<i>JSON</i>	p.12, 13, 14, 14	<i>SGBDRO</i>	p.37
<i>Collection</i> p.41, 44, 45, 51, 55, 67, 68		<i>LDD</i>	p.64, 73	<i>Syntaxe</i>	p.85, 90, 91
<i>Composition</i>	p.45	<i>Méta-langage</i>	p.	<i>TABLE</i>	p.55
<i>Constructeur</i>	p.51, 54	<i>Méthode</i>	p.38, 45, 51, 66	<i>Table imbriquée</i>	p.41, 44, 45, 51, 55
<i>Contrainte</i>	p.64, 73	<i>Modèle</i>	p.37	<i>Type</i>	p.38, 39, 44, 45, 51, 64, 66, 70, 73,
<i>CREATE TABLE</i>	p.64, 73	<i>N:M</i>	p.67, 68	79	
<i>CREATE TYPE</i>	p.39	<i>Nested model</i>	p.41	<i>Valide</i>	p.92
<i>CSV</i>	p.	<i>NESTED TABLE</i>	p.44, 51, 55	<i>Vue</i>	p.
<i>Dénormalisation</i>	p.	<i>Normalisation</i>	p.35	<i>XML</i>	p.84