Bases de données avancées

Master 1 Informatique
2018-2019

Jérôme Darmont
http://eric.univ-lyon2.fr/~jdarmont/

Actualité du cours

http://eric.univ-lyon2.fr/~jdarmont/?page_id=3142

http://eric.univ-lyon2.fr/~jdarmont/?feed=rss2

https://twitter.com/darmont_lyon2 #bda

Objectifs du cours

- SQL : langage de requêtes (bases de données relationnelles)
  - Standard
  - Optimisateurs de requêtes
  - Non procédural
- Programmation nécessaire pour :
  - Tâches complexes
  - Interfaces utilisateurs

Langage PL/pgSQL

- 80 % des données sont peu ou pas structurées
  - Description via le langage XML

Langage XQuery

Partie 1

PL/pgSQL

Procedural Language/PostgreSQL Structured Query Language

Plan

- Introduction
- Bases du langage
- Curseurs
- Gestion des erreurs
- Déclencheurs
- SQL dynamique

Requêtes SQL dans un programme

- SQL encapsulé : Requêtes SQL incorporées dans le code source (PL/SQL, T-SQL, PL/pgSQL, Pro*C…)
- API : Requêtes SQL via des fonctions du langage (Java Persistence API, PHP Data Objects…)
- Interfaces de niveau appel : intergiciel entre le langage et le SGBD (ODBC, JDBC, ADO…)
- Procédures stockées : Fonctions SQL stockées dans la base de données et exécutées par le SGBD (écrites en PL/SQL, T-SQL, PL/pgSQL)

C U R S E U R S

Choix de PostgreSQL

- SGBD relationnel-objet
- Licence libre (BSD)
- Le plus conforme au standard SQL
- Disponible sur de nombreuses plateformes
  - Divers UNIX, dont Linux et Mac OS
  - Windows

Historique de PostgreSQL

- 1974 : Ingres
  (INteractive Graphics REtrieval System)
- 1985 : Postgres (post-Ingres)
- 1995 : Postgres95 (fonctionnalités SQL)
- 1996 : PostgreSQL (v6)
- 2018 : v10.5

Michael Stonebraker

fr.wikipedia.org

Caractéristiques de PostgreSQL

- Types structurés (enregistrements, tableaux) possibles dans les tables
- Comportement stable
- Langage procédural PL/pgSQL proche du PL/SQL d'Oracle (permet le SQL dynamique)
- Interfaçage possible avec des modules externes d'autres langages (PERL, Python…)

Caractéristiques de PL/pgSQL

- Utilisation des types de données, opérateurs et fonctions de SQL
- Stockage du code dans la base de données
  - Sécurité liée à celle du SGBD et de ses droits d'accès
- Exécution au sein du serveur de BD
  - Pas d'allers-retours entre client et serveur ⇒ Performance

Sondage express

A. Jusqu'ici, tout va bien.

B. Je suis déjà perdu.

N° de la question : 835

Répondre sur http://toreply.univ-lille1.fr

Question n° 835

Plan

- ✓ Introduction
- Bases du langage
- Curseurs
- Gestion des erreurs
- Déclencheurs
- SQL dynamique

Structuration en blocs

```
[DECLARE
    -- Déclarations]
BEGIN
    -- Instructions PL/pgSQL
[EXCEPTION
    -- Gestion des erreurs]
END
```


Variables et constantes

- Déclaration dans la section DECLARE d'un bloc PL/pgSQL
- Variables
  ex.
```
dateNaissance DATE;
compteur INTEGER := 0;          -- Initialisation
compteur2 INTEGER DEFAULT 0;    -- Valeur par défaut
id CHAR(5) NOT NULL := 'AP001';
```
- Constantes
  ex.
```
tauxTVA CONSTANT REAL := 0.2;
```


Principaux types de données

Type	Description
boolean, bool	Booléen (vrai/faux)
smallint, int2	Entier de -32768 à +32767
integer, int, int4	Entier de -2147483648 à +2147483647
bigint, int8	Entier de -9223372036854775808 à +9223372036854775807
real, float4	Réel simple précision (6 décimales)
double precision, float8	Réel double précision (15 décimales)
numeric, numeric(P, S), decimal, decimal (P, S)	Nombre jusqu'à 131072 chiffres avant la virgule (P), 16383 après (S)
character[N], char[N]	Chaîne de caractères de longueur N fixe
varchar, varchar[N]	Chaîne de caractères de longueur variable (maximum N si précisé)
text	Texte long de longueur variable (nécessite des opérateurs spécifiques)
date	Date (ex. 'jj/mm/aaaa')
time	Heure (ex. 'hh:mm:ss')
timestamp	Date et heure (ex. 'aaaa-mm-jj hh:mm:ss')

Référencer un type existant

- Type d'une autre variable
  ex.
```
credit REAL;
debit credit%TYPE;
```
- Type de l'attribut d'une table
  ex.
```
numEmp EMP.EMPNO%TYPE;
```
- Type des n-uplets d'une table
  ex.
```
unEtudiant STUDENT%ROWTYPE;
```
- N-uplet indéfini (enregistrement)
  ex.
```
resultat RECORD;
```

À utiliser au maximum !

Types structurés

- Tableaux
  ex. notes NUMERIC(2, 2)[];
  matrice INTEGER[][];
- Types composites (enregistrements)
  ex.
```
CREATE TYPE tEmploye AS (
    num NUMERIC,
    nom VARCHAR
); -- à définir hors de PL/pgSQL
unEmploye tEmploye;
```


Affectation simple

- Variables
  ex.
```
n := 0;
n := n + 1;
```
- Tableaux
  ex.
```
notes := ARRAY[10.2, 13.3, 15.5, 9.8];
matrice := ARRAY[ ARRAY[4, 2],
                  ARRAY[1, 9] ];
```
- Enregistrements
  ex.
```
unEmploye := (1501, 'DARMONT');
```


Lecture de valeurs de la base de données

- Variables
  ex.
```
SELECT custname INTO nomClient
FROM customer WHERE custnum = 10;

SELECT ename, sal INTO nom, salaire
FROM emp WHERE empno = 5000;
```
- Enregistrements
  ex.
```
SELECT empno, ename INTO unEmploye
FROM emp WHERE empno = 1501;

SELECT * INTO resultat
FROM customer where custnum = 20;
```


Opérateurs arithmétiques et logiques

- Opérateurs arithmétiques + - / * **
- Opérateur de concaténation ||
- Opérateurs de comparaison = < > <= >= <>
  IS NULL LIKE BETWEEN IN
- Opérateurs logiques AND OR NOT

Tests (1/3)

IF-THEN, IF-THEN-ELSE ou IF-THEN-ELSIF

```
IF condition1 THEN
        -- Instructions PL/pgSQL
[ELSIF condition2 THEN
        -- Instructions PL/pgSQL]
[ELSE
        -- Instructions PL/pgSQL]
END IF;
```


Tests (2/3)

CASE simple

```
CASE variable
        WHEN val1 THEN          -- Instructions PL/pgSQL
        WHEN val2, val3 THEN    -- Instructions PL/pgSQL
        WHEN val4 THEN          -- Instructions PL/pgSQL
        [ELSE                   -- Instructions par défaut]
END CASE;
```


Tests (3/3)

CASE par intervalles

```
CASE
    WHEN var BETWEEN val1 AND val2 THEN
            -- Instructions PL/pgSQL
    WHEN var BETWEEN val2 + 1 AND val3 THEN
            -- Instructions PL/pgSQL
END CASE;
```


Boucles

- Pour

```
FOR iterateur IN [REVERSE] min..max [BY pas] LOOP
    -- Instructions PL/pgSQL
END LOOP;
```

- Tant que

```
WHILE condition LOOP
    -- Instructions PL/pgSQL
END LOOP;
```

- Répéter jusqu'à

```
LOOP
    -- Instructions PL/pgSQL
    EXIT WHEN condition;
END LOOP;
```

Boucles sur tableaux

- Tableau

```
FOREACH note IN ARRAY notes LOOP      -- note est un NUMERIC(2, 2)
    -- Instructions PL/pgSQL
END LOOP;
```

- Matrice

```
FOREACH n IN ARRAY matrice LOOP       -- n est un INTEGER
    -- Instructions PL/pgSQL
END LOOP;
-- Et oui, pas de boucles imbriquées !
```


Quizz

Combien y a-t-il de sections dans un bloc PL/pgSQL ?

A. 1
B. 2
C. 3
D. 4
E. 5

N° de la question : 472

Répondre sur http://toreply.univ-lille1.fr

Question n° 472

Implémentation d'un bloc

- Dans une fonction

```
ex. CREATE [OR REPLACE] FUNCTION test() RETURNS VOID AS $$
        -- bloc PL/pgSQL
    $$ LANGUAGE plpgsql;
```

- Exécution de la fonction

```
ex. SELECT test();          -- Données brutes
    SELECT * FROM test();   -- Sous forme de table
```


Exemple de fonction

```
-- Calcul de prix TTC
CREATE OR REPLACE FUNCTION calculPrixTTC(idProduct NUMERIC) RETURNS REAL AS $$
    DECLARE
        tauxTVA CONSTANT REAL := 0.2;
        prixHT demo_product_info.list_price%TYPE;
    BEGIN
        -- Lire le prix HT
        SELECT list_price INTO prixHT FROM demo_product_info;
            WHERE product_id = idProduct;
        -- Retourner le prix TTC
        RETURN prixHT * (1 + tauxTVA) ::REAL;
    END
$$ LANGUAGE plpgsql;

-- Exécution
SELECT calculPrixTTC(10);
```

Fonction retournant plusieurs valeurs

Paramètres de sortie

```
CREATE OR REPLACE FUNCTION calculs(n1 INT, n2 INT, OUT somme INT, OUT produit INT) AS $$
                                                        -- Pas de RETURN
    BEGIN
        somme := n1 + n2;
        produit := n1 * n2;
    END
$$ LANGUAGE plpgsql;

SELECT calculs(4, 5);              -- Sous forme d'enregistrement (valeurs uniquement)
-- ou
SELECT * FROM calculs(4, 5);       -- Sous forme de table (avec entêtes somme et produit)
```


Fonction retournant plusieurs enregistrements

```
CREATE OR REPLACE FUNCTION serie(taille INT, pas INT) RETURNS SETOF INT AS $$
    DECLARE
        i INT;
    BEGIN
        FOR i IN 1..taille BY pas LOOP
            RETURN NEXT i;
        END LOOP;
        RETURN;
    END
$$ LANGUAGE plpgsql;

SELECT serie(10, 2);
```


Appel de fonction dans une fonction

```
CREATE OR REPLACE FUNCTION droleDeDiv(n1 INT, n2 INT) RETURNS REAL AS $$
    DECLARE
        s INT;
        p INT;
    BEGIN
        SELECT * INTO s, p FROM calculs(n1, n2);
        RETURN  p / s ::REAL;
    END
$$ LANGUAGE plpgsql;

SELECT droleDeDiv(5, 6);
```


Récursivité

```
CREATE OR REPLACE FUNCTION facto(n INTEGER) RETURNS INTEGER AS $$
    DECLARE
        f INTEGER;
    BEGIN
        IF n = 1 THEN -- Condition d'arrêt
            RETURN 1;
        ELSE
            SELECT * INTO f FROM facto(n - 1); -- Appel récursif
            RETURN n * f;
        END IF;
    END
$$ LANGUAGE plpgsql;

SELECT facto(10);
```

Sondage express

N° de la question : 138

Peut-on écrire un bloc PL/pgSQL en-dehors d'une fonction?

Répondre sur http://toreply.univ-lille1.fr

Question n° 138

Plan

- ✓ Introduction
- ✓ Bases du langage
- Curseurs
- Gestion des erreurs
- Déclencheurs
- SQL dynamique

Contexte d'utilisation des curseurs

- Requête qui retourne un seul n-uplet
  - SELECT INTO
  - Stockage du résultat dans une ou plusieurs variables ou un enregistrement
- Requête qui retourne plusieurs n-uplets
  - Nécessité d'un curseur
  - Structure de données en mémoire qui stocke le résultat de la requête (≈ tableau d'enregistrements)

Curseur implicite (non lié)

```
-- Parcours complet du curseur
CREATE OR REPLACE FUNCTION listeEmp() RETURNS SETOF tEmploye  AS $$
      DECLARE
            nuplet emp%ROWTYPE;
            e tEmploye;
      BEGIN
            FOR nuplet IN SELECT * FROM emp LOOP
                  e.num := nuplet.empno;
                  e.nom := LOWER(nuplet.ename);
                  RETURN NEXT e;
            END LOOP;
            RETURN;
      END
$$ LANGUAGE plpgsql;

SELECT * FROM listeEmp();
```

Curseur explicite (lié)

```
-- Parcours ad-hoc du curseur
CREATE OR REPLACE FUNCTION listeEmp2() RETURNS SETOF tEmploye  AS $$
    DECLARE
        cursEmp CURSOR FOR SELECT * FROM emp;
        nuplet emp%ROWTYPE;
        e tEmploye;
    BEGIN
        OPEN cursEmp;
        FETCH cursEmp INTO nuplet;          -- Lecture du 1er n-uplet
        WHILE FOUND LOOP
            e.num := nuplet.empno;
            e.nom := LOWER(nuplet.ename);
            RETURN NEXT e;
            FETCH cursEmp INTO nuplet;      -- Lecture du n-uplet suivant
        END LOOP;
        CLOSE cursEmp;
        RETURN;
    END
$$ LANGUAGE plpgsql;
```


Curseur explicite (lié)

```
-- Parcours vraiment ad-hoc du curseur (renvoie 1 résultat sur 3)
CREATE OR REPLACE FUNCTION listeEmp3(pas INT) RETURNS SETOF tEmploye  AS $$
    DECLARE -- Pas de changement
    BEGIN
        OPEN cursEmp;
        FETCH cursEmp INTO nuplet;
        WHILE FOUND LOOP
            e.num := nuplet.empno;
            e.nom := LOWER(nuplet.ename);
            RETURN NEXT e;
            MOVE FORWARD pas FROM cursEmp;    -- MOVE cursEmp; pour un seul décalage
            FETCH cursEmp INTO nuplet;
        END LOOP;
        CLOSE cursEmp;
        RETURN;
    END
$$ LANGUAGE plpgsql;

SELECT * FROM listeEmp3(2);
```


Curseur paramétré

```
CREATE OR REPLACE FUNCTION listeEmp4(salPlancher DECIMAL) RETURNS SETOF tEmploye  AS $$
    DECLARE
        cursEmp CURSOR(salMin DECIMAL) FOR SELECT * FROM emp WHERE sal >= salMin;
        nuplet emp%ROWTYPE;
        e tEmploye;
    BEGIN
        OPEN cursEmp(salPlancher);
        FETCH cursEmp INTO nuplet;
        WHILE FOUND LOOP
            e.num := nuplet.empno;
            e.nom := LOWER(nuplet.ename);
            RETURN NEXT e;
            FETCH cursEmp INTO nuplet;
        END LOOP;
        CLOSE cursEmp;
        RETURN;
    END
$$ LANGUAGE plpgsql;

SELECT * FROM listeEmp4(2000);
```


Quizz

N° de la question : 507

On veut appliquer un échantillonnage sur les n-uplets d'une table. Quel type de curseur doit-on utiliser pour les sélectionner ?

A. Curseur implicite
B. Curseur explicite
C. Curseur paramétré

Répondre sur http://toreply.univ-lille1.fr

Question n° 507

Plan

- Introduction
- Bases du langage
- Curseurs
- Gestion des erreurs
- Déclencheurs
- SQL dynamique

Signalement d'erreur

RAISE niveauErreur 'message';

Niveau d'erreur	Priorité	Description
DEBUG	1	Écrit le message dans le log
LOG	2	Écrit le message dans le log
INFO	3	Écrit le message dans le log et l'envoie au client
NOTICE	4	Écrit le message dans le log et l'envoie au client
WARNING	5	Écrit le message dans le log et l'envoie au client
EXCEPTION	6	Interrompt la transaction courante

Exceptions personnalisées

```
CREATE OR REPLACE FUNCTION testErreur1(noDept INTEGER) RETURNS REAL AS $$
    DECLARE
        nbEmp INTEGER;
        ref CONSTANT INTEGER := 85;
    BEGIN
        SELECT COUNT(*) INTO nbEmp FROM emp WHERE deptno = noDept;
        IF nbEmp = 0 THEN
            RAISE EXCEPTION 'Pas d''employé dans ce département';
            -- RAISE EXCEPTION 'Pas d''employé dans le département %', noDept;
        END IF;
        RETURN  ref / nbEmp ::REAL;
    END
$$ LANGUAGE plpgsql;

SELECT * FROM testErreur1(99);
```


Exploitation des exceptions systèmes

```
CREATE OR REPLACE FUNCTION testErreur2(noDept INTEGER) RETURNS REAL AS $$
    DECLARE
        nbEmp INTEGER;
        ref CONSTANT INTEGER := 85;
    BEGIN
        SELECT COUNT(*) INTO nbEmp FROM emp WHERE deptno = noDept;
        RETURN ref / nbEmp ::REAL;
    EXCEPTION
        WHEN division_by_zero THEN
            RAISE WARNING 'Pas d''employé dans le département %', noDept;
            RETURN NULL;
    END
$$ LANGUAGE plpgsql;

SELECT * FROM testErreur2(99);
```

Traitement par défaut

```
CREATE OR REPLACE FUNCTION testErreur3(noDept INTEGER) RETURNS REAL AS $$
    DECLARE
        nbEmp INTEGER;
        ref CONSTANT INTEGER := 85;
    BEGIN
        IF noDept <= 0 THEN
            RAISE EXCEPTION 'noDept ne peut pas être négatif';
        END IF;
        SELECT COUNT(*) INTO nbEmp FROM emp WHERE deptno = noDept;
        RETURN ref / nbEmp ::REAL;
    EXCEPTION
        WHEN division_by_zero THEN
            RAISE WARNING 'Personne dans le département %', noDept; RETURN NULL;
        WHEN others THEN -- On peut aussi remplacer others par raise_exception
            RETURN -1;      -- Code d'erreur
  END
$$ LANGUAGE plpgsql;
```


Quelques autres exceptions système

Code	Nom
22004	null_value_not allowed
22003	numeric_value_out_of_range
22012	division_by_zero
23503	foreign_key_violation
23505	unique_violation
28P01	invalid_password
42501	insufficient_privilege
42883	undefined column
54011	too_many_columns
P0002	no_data_found
P0003	too_many_rows

Plan

- ✓ Introduction
- ✓ Bases du langage
- ✓ Curseurs
- ✓ Gestion des erreurs
- Déclencheurs
- SQL dynamique

Définition des déclencheurs

- Définition : Fonction associée à une table et exécutée automatiquement lorsque des événements liés à des actions sur la table surviennent (mises à jour, principalement).
- Complètent les contraintes d'intégrité en permettant de créer des règles d'intégrité complexes.
  - Éléments des bases de données actives.

Principaux types de déclencheurs

	Insert	Delete	Update
Before	1	2	3
After	4	5	6

Création/suppression de déclencheur

```
CREATE TRIGGER nomDeclencheur
    BEFORE | AFTER
    INSERT | DELETE | UPDATE | [INSERT] [[OR] DELETE] [[OR] UPDATE]
    ON nomTable
    FOR EACH ROW | FOR EACH STATEMENT
    EXECUTE PROCEDURE nomFonction();

DROP TRIGGER nomDeclencheur
    ON nomTable;
```


Variables systèmes des déclencheurs

- NEW : Enregistrement contenant le n-uplet inséré ou modifié

  Ex. INSERT INTO client VALUES (1, 'NouveauClient');

  NEW.NumCli prend la valeur 1 dans le déclencheur.
  NEW.Nom prend la valeur 'NouveauClient' dans le déclencheur.

- OLD : Enregistrement contenant l'ancien n-uplet supprimé ou modifié

  Ex. DELETE FROM client WHERE NumCli = 33;

  OLD.NumCli prend la valeur 33 dans le déclencheur.

Exemple de déclencheur (1/2)

```
-- Emulation de clé primaire sur la table EMP

CREATE OR REPLACE FUNCTION checkEmpPK() RETURNS TRIGGER AS $$
    DECLARE
        n INTEGER;
    BEGIN
        -- La clé est-elle vide ?
        IF NEW.empno IS NULL THEN
            RAISE EXCEPTION 'La clé primaire doit avoir une valeur !';
        END IF;
```

Exemple de déclencheur (2/2)

```
        -- La clé existe-t-elle déjà ?
        SELECT COUNT(empno) INTO n FROM emp
            WHERE empno = NEW.empno;
        IF n > 0 THEN
            RAISE EXCEPTION 'Clé primaire déjà utilisée !';
        END IF;
        RETURN NEW;
    END
$$ LANGUAGE plpgsql;


CREATE TRIGGER trigEmpPK
    BEFORE INSERT OR UPDATE ON emp
    FOR EACH ROW EXECUTE PROCEDURE checkEmpPK();
```


Quizz

N° de la question : 723

Deux tables T et T' ont la même structure.
Chaque ajout de n-uplet dans T doit être répercuté dans T'.
Quel type de déclencheur faut-il utiliser ?

A. BEFORE INSERT FOR EACH STATEMENT
B. BEFORE INSERT FOR EACH ROW
C. AFTER INSERT FOR EACH STATEMENT
D. AFTER INSERT FOR EACH ROW

Répondre sur http://toreply.univ-lille1.fr

Question n° 723

Plan

- ✓ Introduction
- ✓ Bases du langage
- ✓ Curseurs
- ✓ Gestion des erreurs
- ✓ Déclencheurs
- SQL dynamique

SQL statique *vs.* SQL dynamique

- Exemples
  - Fonction qui met la table EMP à jour
    ⇒ SQL statique (la requête est connue à la compilation)
  - Fonction qui met à jour une table dont le nom est un paramètre
    ⇒ SQL dynamique (la requête complète n'est pas connue à la compilation)
- Définition du SQL dynamique :
  Construction d'une requête SQL à la volée dans un bloc PL/pgSQL

Requête dynamiques

- Exécution : EXECUTE requete [INTO resultat];
  - requete est une chaîne de caractères
  - resultat est une variable, un ensemble de variables ou un enregistrement
- Notes
  - Requêtes paramétrées : valeurs de la base de données. Si l'on veut paramétrer des objets (tables, vues…), requête dynamique.
  - Requêtes qui modifient la structure de la base de données (CREATE, DROP, ALTER…), même statiques : doivent être exécutées en mode dynamique.

Exemple de requête dynamique

```
CREATE OR REPLACE FUNCTION tailleTable(nomTable VARCHAR) RETURNS INTEGER AS $$
        DECLARE
                n INTEGER;
        BEGIN
                EXECUTE 'SELECT COUNT(*) FROM ' || nomTable INTO n;
        RETURN n;
    END
$$ LANGUAGE plpgsql;

SELECT tailleTable('EMP');
```


Curseurs dynamiques (1/2)

```
-- Exemple 1
CREATE OR REPLACE FUNCTION parcoursTable(nomTable VARCHAR) RETURNS VOID AS $$
        DECLARE
                dynCurs REFCURSOR;
                nuplet RECORD;
        BEGIN
                OPEN dynCurs FOR EXECUTE 'SELECT * FROM ' || nomTable;
                FETCH dynCurs INTO nuplet;
                WHILE FOUND LOOP
                        -- Opérations sur nuplet
                        FETCH dynCurs INTO nuplet;
                END LOOP;
                CLOSE dynCurs;
        END
$$ LANGUAGE plpgsql;

SELECT parcoursTable('EMP');
```

Une fonction ne peut pas retourner un SETOF RECORD.

Curseurs dynamiques (2/2)

```
-- Exemple 2
CREATE FUNCTION contenuTable(nomTable VARCHAR, nomID VARCHAR) RETURNS SETOF VARCHAR AS $$
        DECLARE
                dynCurs REFCURSOR;
                id VARCHAR;
        BEGIN
                OPEN dynCurs FOR EXECUTE 'SELECT ' || nomID || ' FROM ' || nomTable;
                FETCH dynCurs INTO id;
                WHILE FOUND LOOP
                        RETURN NEXT id;
                        FETCH dynCurs INTO id;
                END LOOP;
                CLOSE dynCurs;
                RETURN;
        END
$$ LANGUAGE plpgsql;

SELECT contenuTable('EMP', 'EMPNO');
```

Quizz

N° de la question : 106

Lesquelles de ces requêtes sont dynamiques ?

A. SELECT * FROM emp;
B. SELECT * FROM emp WHERE empno = n;
C. SELECT COUNT(*) FROM nom_table;
D. DROP TABLE emp;

Répondre sur http://toreply.univ-lille1.fr

Question n° 106

Plan

- Introduction
- Bases du langag
- Curseurs
- Gestion des e urs
- Déclencheurs
- SQL dynamique

Partie 2

XML/XQuery

Plan

- Introduction
- Documents XML
- Langage XQuery
  - XPath
  - Requêtes FLWOR
  - Requêtes complexes

Données structurées

- Données organisées en entités
- Entités similaires : forment des groupes (classes)
- Entités du même groupe : même description (attributs)
- Pour toutes les entités d'un groupe :
  - Chaque attribut a le même type
  - Chaque valeur d'attribut a la même taille
  - Tous les attributs sont présents
  - Les attributs sont toujours dans le même ordre
- Données structurés : décrites par un schéma
  - Généralement stockées dans des bases de données

Données non structurées

- Données de tous types
- Données qui ne suivent aucun schéma ni séquence prédéterminé
- Données qui ne suivent aucune règle
- Données qui ne sont pas prévisibles
- Exemples de données non-structurées :
  - Textes
  - Images
  - Vidéos
  - Sons

Données semi-structurées

Données semi-structurées

- Données organisées en entités sémantiques
- Entités similaires : groupes

Structurées

- Entités du même groupe : peuvent ne pas avoir les mêmes attributs
- Pour toutes les entités d'un groupe :
  - Un même attribut peut avoir des types différents
  - Une même valeur d'attribut peut avoir des tailles différentes
  - Des attributs peuvent être manquants ou dupliqués
  - L'ordre des attributs n'est pas nécessairement important

Non-structurées

- Données semi-structurées : autodescriptives
  - Pages web, documents XML, courriels…

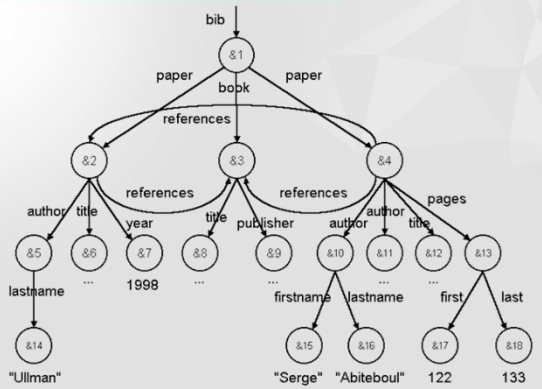
Exemple de données semi-structurées

- Nom — Jérôme Darmont
- Courriel — jerome.darmont@univ-lyon2.fr, jdarmont@eric.univ-lyon2.fr
- Courriel — sabine.loudcher@univ-lyon2.fr
- Nom
  - Prénom — Loudcher
  - Nom de famille — Sabine
- Nom — Julien Velcin
- Affiliation — Université Lumière Lyon 2

Modèle de données semi-structuré

- Avantages
  - Peut représenter des informations issues de sources de données qui ne peuvent pas êtres contraintes par un schéma
  - Format flexible pour l'interopérabilité
  - Permet de voir des données structurées comme semi-structurées (Web)
  - Schéma facilement évolutif
- Inconvénients
  - Performance des requêtes sur données à grande échelle
- Représentations standards
  - Electronic Data Interchange (EDI) : domaine financier
  - Object Exchange Model (OEM) : modèle basé sur les graphes
  - SGML, HTML et XML

Exemple de graphe OEM



Gestion de données semi-structurées

- Modélisation des données semi-structurées
  - Graphes (OEM) – Modèle logique
  - XML – Modèle physique
- Requêtage des données semi-structurées
  - XPath
  - XQuery
- Stockage des données semi-structurées
  - Fichiers plats
  - Bases de données relationnelles, relationnelles-objets ou natives XML

Références

- Peter Wood, Birkbeck University of London
  Semi-Structured Data
  http://www.dcs.bbk.ac.uk/~ptw/
- Mike Bergman, Structured Dynamics LLC
  Semi-structured Data: Happy 10th Birthday!
  http://www.mkbergman.com/153/semi-structured-data-happy-10th-birthday/

Sondage express

N° de la question : 460

Pensez-vous avoir compris la Différence entre données structurées, non structurées et semi-structurées ?

Répondre sur http://toreply.univ-lille1.fr

Question n° 460

Pourquoi pas JSON, YAML et les BD NoSQL ?

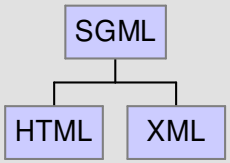
- SGBD NoSQL type MongoDB : plus efficaces que XML/XQuery si besoin de stockage distribué
- Pour l'*analytics*, Xquery est plus puissant (opérateur // inexistant dans MongoDB)
- Les SGBD NoSQL sont étudiés en M2 ! 😄

Plan

- ✓ Introduction
- Documents XML
- Langage XQuery
  - XPath
  - Requêtes FLWOR
  - Requêtes complexes

Le langage XML

- XML : Extensible Markup Language
  - Format de structuration de données et de documents Internet issu de SGML
  - Définition, gestion, création, transmission et partage de documents
- XML est un standard du W3C
  - 1996 : Brouillon
  - 1997 : XML 1.0
  - 2004 : XML 1.1



Exemple de document XML

```
<?xml version="1.1"  encoding="utf-8" ?> <!-- Prologue (obligatoire) -->
<annuaire_professeurs> <!-- Élément racine -->
   <!– Sous-éléments  -->
   <professeur>
      <nom>Jérôme Darmont</nom>
      <courriel>jerome.darmont@univ-lyon2.fr</courriel>
      <cours>Bases de données avancées</cours>
      <cours>Programmation Web</cours>
   </professeur>
   <professeur>
      <nom>Julien Velcin</nom>
      <courriel>julien.velcin@univ-lyon2.fr</courriel>
      <cours>Programmation orientée objet</cours>
   </professeur>
   <!-- Etc. -->
</annuaire_professeurs> <!-- Balise de fin -->
```

Ensemble d'éléments imbriqués matérialisés par des balises

Règles d'écriture d'un document XML (1/2)

- Un document XML a un et un seul élément racine.
- Les éléments doivent être correctement emboîtés (les balises ouvrantes et fermantes ne doivent pas se chevaucher).
- Tout élément doit avoir une balise ouvrante et une balise fermante.
- Le nom d'un élément doit être identique dans la balise ouvrante et la balise fermante.
- Les noms d'éléments sont sensibles à la casse. Ils doivent commencer par une lettre ou par _ suivi(e) de lettres, de chiffres, de . , de – ou de _.

Règles d'écriture d'un document XML (2/2)

- Les noms d'éléments commençant par XML (dans toutes combinaisons de minuscules et majuscules) sont réservés à des fins de standardisation.
- Un document XML respectant ces règles est dit bien formé.
- Un document XML doit être bien formé !
- Un document XML peut de plus être valide s'il se conforme à la structure définie dans une DTD ou un Schéma XML.

Document Type Definition XML Schema

Schémas XML

DTD	XML Schema
Syntaxe spécifique (non XML)	Exprimé dans un document XML
Typage faible	Typage fort
Modélisation partielle impossible	Modélisation partielle possible
Interprétable par un·e utilisateur·trice humain·e	Conçu pour des traitements automatiques

Éléments XML (1/2)

- Caractères non-autorisés : < &]]>
- Eléments emboîtés : profondeur non limitée
  - Ex.
```
<annuaire_professeurs>
    <professeur>
        <nom>
            <nom_famille>Zighed</nom_famille>
            <prenom>Abdelkader</prenom>
            <prenom>Djamel</prenom>
        </nom>
    </professeur>
</annuaire_professeurs>
```


Éléments XML (2/2)

- Section CDATA : Bloc de texte libre dans lequel seule la chaîne]]> est interdite
  - Ex.
```
<nom>
    <![CDATA[<Darmont> & <Loudcher>]]>
</nom>
```
- Élément vide : sans contenu
  - Ex. <courriel></courriel>
  - Formulation équivalente <courriel />

Attributs d'éléments XML

- Attributs : données associées à un élément, complémentaires du contenu
- Définition : couple nom/valeur dans la balise ouvrante de l'élément
  - Ex. <bureau campus="PdA" batiment="K">063</bureau>

attribut valeur attribut valeur

- Les attributs sont possibles dans les éléments vides.
  - Ex. <image source="ma_bobine.png" />

Contenu d'élément XML *vs.* attributs (1/2)

- Que choisir ?
  - <professeur>
    <nom>Darmont</nom>
    </professeur>
  - <professeur nom="Darmont" />
- 4 principes pour décider
  - Uche Ogbuji, Fourthought, Inc.
  - Source : http://www.ibm.com/developerworks/xml/library/x-eleatt/

Contenu d'élément XML *vs.* attributs (2/2)

Principe	Élément	Attribut
Contenu principal	Information essentielle	Information périphérique
Information structurée	Information hiérarchisée	Information atomique
Lisibilité	Utilisateur.trice humain.e	Traitement automatique
Relation élément/attribut	Information précisée par une autre	

Exemple de relation élément/attribut

```
<stock>
    <produit quantité="1500">
        <nom>Ordinateur</nom>
    </produit>
    <produit quantité="500">
        <nom>Imprimante</nom>
    </produit>
</stock>
```


Quizz

Dans un document XML, les données sont stockées dans :

A. Les éléments

B. Les attributs

N° de la question : 141

Répondre sur http://toreply.univ-lille1.fr

Question n° 141

Plan

- ✓ Introduction
- ✓ Documents XML
- Langage XQuery
  - XPath
  - Requêtes FLWOR
  - Requêtes complexes

Le langage XQuery

- Langage de requêtes pour données XML
- Similarités avec SQL
- Conçu par le W3C
- Basé sur des expressions XPath (mêmes modèle de données, fonctions, opérateurs)
- Versions
  - 2007 : XQuery 1.0 ⊇ XPath 2.0
  - 2017 : XQuery 3.1 ⊇ XPath 3.1
- Standardisation en cours
- Soutenu par les éditeurs de SGBD (Oracle, Microsoft, IBM…)

Document XML exemple (1/2)

```
<?xml version="1.1" encoding="utf-8" ?>
<catalogue>
    <dvd zone="1">
        <titre>Blade runner</titre>
        <realisateur>Ridley Scott</realisateur>
        <annee>1982</annee>
        <langue>Anglais</langue>
        <prix>14.79</prix>
    </dvd>
    <dvd zone="2">
        <titre>La grande vadrouille</titre>
        <realisateur>Gérard Oury</realisateur>
        <annee>1966</annee>
        <duree>122</duree>
        <langue>Français</langue>
        <prix>19.82</prix>
    </dvd>                                    <!-- (…) -->
```


Document XML exemple (2/2)

```
    <dvd zone="2">
        <titre>Le fabuleux destin d'Amélie Poulain</titre>
        <realisateur>Jean-Pierre Jeunet</realisateur>
        <annee>2001</annee>
        <duree>120</duree>
        <langue>Français</langue>
        <prix>14.99</prix>
    </dvd>
    <dvd zone="2">
        <titre>The big Lebowski</titre>
        <realisateur>Ethan Coen</realisateur>
        <realisateur>Joel Coen</realisateur>
        <annee>1997</annee>
        <duree>112</duree>
        <langue>Français</langue>
        <langue>Anglais</langue>
        <prix>19.82</prix>
    </dvd>
</catalogue>
```


Plan

- ✓ Introduction
- ✓ Documents XML
- ✓ Langage XQuery
  - XPath
  - Requêtes FLWOR
  - Requêtes complexes

Expressions de chemins (1/3)

- Document XML entier
  doc("dvd.xml")/catalogue
  Résultat
  Tout le document

Par l'exemple

- Un élément donné
  doc("dvd.xml")/catalogue/dvd
  doc("dvd.xml")/catalogue/dvd/titre
  Résultat
  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>

Expressions de chemins (2/3)

- Un attribut donné
  doc("dvd.xml")/catalogue/dvd/data(@zone)
  Résultat
  1 2 2 2

- Un élément donné quel que soit son niveau hiérarchique
  doc("dvd.xml")/catalogue//titre
  //titre
  Résultat
  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>

Expressions de chemins (3/3)

- Tous les sous-éléments d'un élément
  doc("dvd.xml")/catalogue/dvd/*
  Résultat
  <titre>Blade runner</titre>
  <realisateur>Ridley Scott</realisateur>
  <annee>1982</annee>
  <duree>117</duree>
  <langue>English</langue>
  <prix>14.79</prix>
  <titre>La grande vadrouille</titre>
  <realisateur>Gérard Oury</realisateur>
  <annee>1966</annee>
  <duree>122</duree>
  <langue>French</langue>
  <prix>19.82</prix>

Etc.

Prédicats XPath (1/2)

- i^e, dernier, i premiers/derniers éléments
  doc("dvd.xml")/catalogue/dvd[1]
  doc("dvd.xml")/catalogue/dvd[last()]
  doc("dvd.xml")/catalogue/dvd[position() < 3]/titre
  Résultat
  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>

- Éléments possédant un sous-élément ou attribut donné
  doc("dvd.xml")/catalogue/dvd[duree]/titre
  Résultat
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>
  doc("dvd.xml")/catalogue/dvd[@zone]

Prédicats XPath (2/2)

- Condition sur un élément ou un attribut

```
doc("dvd.xml")/catalogue/dvd[prix < 15]
doc("dvd.xml")/catalogue/dvd[@zone = "2" and prix < 15]/titre
```

Résultat

```
<titre>Le fabuleux destin d'Amélie Poulain</titre>
```

- Combinaison de chemins

```
doc("dvd.xml")//titre | doc("dvd.xml")//prix
```

Résultat

```
<titre>Blade runner</titre><prix>14.79</prix>
<titre>La grande vadrouille</titre><prix>19.82</prix>
<titre>Le fabuleux destin d'Amélie Poulain</titre><prix>14.99</prix>
<titre>The big Lebowski</titre> <prix>19.82</prix>
```


Quizz

N° de la question : 709

Lesquels de ces chemins sont-ils corrects ?

A. /catalogue/dvd/prix
B. doc("dvd.xml")/catalogue/dvd/prix
C. //prix
D. /catalogue//prix

Répondre sur http://toreply.univ-lille1.fr

Question n° 709

Plan

- ✓ Introduction
- ✓ Documents XML
- ✓ Langage XQuery
  - ✓ XPath
  - – Requêtes FLWOR
  - – Requêtes complexes

Requêtes FLWOR

- For, Let, Where, Order by, Return
- Clause For (1/3) : lie une variable à chaque élément retourné par une expression (itération)

Exemple

```
for $x in (1 to 3) <!-- Ceci est un commentaire -->
return <res>{$x}</res>
```

Résultat

```
<res>1</res>
<res>2</res>
<res>3</res>
```

Clause For (2/3)

Exemple

```
for  $x in (1, 2),
     $y in (10, 20) (: Ceci est également un commentaire :)
return <res>x = {$x} et y = {$y}</res>
```

Résultat

```
<res>x = 1 et y = 10</res>
<res>x = 1 et y = 20</res>
<res>x = 2 et y = 10</res>
<res>x = 2 et y = 20</res>
```


Clause For (3/3)

Exemple

```
for $x at $i in doc("dvd.xml")/catalogue/dvd/titre
return <dvd id="{$i}">{data($x)}</dvd>
```

Résultat

```
<dvd id="1">Blade runner</dvd>
<dvd id="2">La grande vadrouille</dvd>
<dvd id="3">Le fabuleux destin d'Amélie Poulain</dvd>
<dvd id="4">The big Lebowski</dvd>
```


Clause Let

- Clause Let : Assigner une ou plusieurs valeurs à une variable (pas d'itération)

Exemple

```
let $x := (1 to 5)
return <res>{$x}</res>
```

Résultat

```
<res>1 2 3 4 5</res>
```


Clause Where

- Clause Where : Spécifie une ou plusieurs conditions sur le résultat

Exemple

```
for $x in doc("dvd.xml")/catalogue/dvd
where $x/prix > 15
return $x/titre
```

Exemple

```
for $x in doc("dvd.xml")/catalogue/dvd
where $x/@zone = "2" and $x/prix < 10
return $x/titre
```

Clauses Order by et Return (1/2)

- Clause Order by : Trie le résultat

  Exemple

```
for $x in doc("dvd.xml")/catalogue/dvd
order by $x/titre
return $x/titre
```

  Exemple

```
for $x in doc("dvd.xml")/catalogue/dvd
order by $x/@zone, $x/titre descending
return $x/titre
```

- Clause Return : Spécifie le résultat

Clause Return (2/2)

- Expressions conditionnelles

  Exemple

```
for $x in doc("dvd.xml")/catalogue/dvd
return   if ($x/@zone="1")
         then <zoneUS>{data($x/titre)}</zoneUS>
         else <zoneEU>{data($x/titre)}</zoneEU>
```

  Résultat

```
<zoneUS>Blade runner</zoneUS>
<zoneEU>La grande vadrouille</zoneEU>
<zoneEU>Le fabuleux destin d'Amélie Poulain</zoneEU>
<zoneEU>The big Lebowski</zoneEU>
```


Fonctions XPath/XQuery (1/2)

- Fonctions d'accès : data()…
- Fonctions numériques : abs(), floor(), ceiling(), round(), number()…
- Fonctions de chaînes : string-length(), upper-case(), lower-case(), normalize-space(), substring(), substring-after(), replace(), contains()…
- Fonctions temporelles : day-from-date(), year-from-date()…
- Fonctions de séquences : exists(), distinct-values(), reverse()…
- Fonctions contextuelles : last(), position()…
- Fonctions booléennes : not()…

Fonctions XPath/XQuery (2/2)

Exemple d'appel à une fonction

```
for $x in doc("dvd.xml")/catalog/dvd/titre
let $titreMAJ := upper-case($x)
return <film>{$titreMAJ}</film>
```

Résultat

```
<film>BLADE RUNNER</film>
<film>LA GRANDE VADROUILLE</film>
<film>LE FABULEUX DESTIN D'AMÉLIE POULAIN</film>
<film>THE BIG LEBOWSKI</film>
```

Plan

- Introduction
- Documents XML
- Langage XQuery
  - XPath
  - Requêtes FLWOR
  - Requêtes complexes

Clause Group by (XQuery 3)

Regroupement sur un critère

```
for $d in /catalogue/dvd
group by $z := $d/@zone
return    <zone value="{$z}">
              <prix_moyen>{avg($d/prix)}</prix_moyen>
          </zone>
```

Regroupement multiple

```
for $d in /catalogue/dvd
group by $z := $d/@zone, $a := $d/annee
return    <groupe zone="{$z}" annee="{$a}">
              <prix_moyen>{avg($d/prix)}</prix_moyen>
          </groupe>
```


Jointures – Documents exemples (1/3)

```
<?xml version="1.1" encoding="utf-8" ?>  <!-- document 1 : clients.xml -->
<clients>
    <client id="1">
        <nom>Loudcher</nom>
        <prenom>Sabine</prenom>
        <addresse>Bureau K073</addresse>
    </client>
    <client id="2">
        <nom>Bentayeb</nom>
        <prenom>Fadila</prenom>
        <addresse>Bureau K061</addresse>
    </client>
    <client id="3">
        <nom>Darmont</nom>
        <prenom>Jérôme</prenom>
        <addresse>Bureau K063</addresse>
    </client>
</clients>
```


Jointures – Documents exemples (2/3)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- document 2 : produits.xml -->
<produits>
    <produit id="10">
        <nom>Ordinateur</nom>
    </produit>
    <produit id="20">
        <nom>Moniteur</nom>
    </produit>
    <produit id="30">
        <nom>Imprimante</nom>
    </produit>
</produits>
```

Jointures – Documents exemples (3/3)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- document 3 : commandes.xml -->
<commandes>
    <commande cli-id="1" prod-id="10">
        <quantite>3</quantite>
    </commande>
    <commande cli-id="1" prod-id="20">
        <quantite>15</quantite>
    </commande>
    <commande cli-id="2" prod-id="10">
        <quantite>7</quantite>
    </commande>
    <commande cli-id="2" prod-id="30">
        <quantite>10</quantite>
    </commande>
    <commande cli-id="3" prod-id="30">
        <quantite>5</quantite>
    </commande>
</commandes>
```


Jointures de documents XML (1/3)

Exemple

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande
where $c/@id = $o/@cli-id
return <res>{data($c/nom)}, {data($c/prenom)}:
    {data($o/quantite)}</res>
```

Résultat

```
<res>Loudcher, Sabine : 3</res>
<res>Loudcher, Sabine : 15</res>
<res>Bentayeb, Fadila : 7</res>
<res>Bentayeb, Fadila : 10</res>
<res>Darmont, Jérôme : 5</res>
```


Jointures de documents XML (2/3)

Exemple

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande,
      $p in doc("produits.xml")//produit
where $c/@id = $o/@cli-id
and $o/@prod-id = $p/@id
return <res>{data($c/nom)}, {data($c/prenom)}:
        {data($o/quantite)} x {data($p/nom)}</res>
```

Résultat

```
<res>Loudcher, Sabine : 3 x Ordinateur</res>
<res>Loudcher, Sabine : 15 x Moniteur</res>
<res>Bentayeb, Fadila : 7 x Ordinateur</res>
<res>Bentayeb, Fadila : 10 x Imprimante</res>
<res>Darmont, Jérôme : 5 x Imprimante</res>
```


Jointures de documents XML (3/3)

Variantes avec les conditions de jointures exprimées en prédicats de chemins

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande[@cli-id=$c/@id]
return <res>{data($c/nom)}, {data($c/prenom)}:
        {data($o/quantite)}</res>

for   $c in //client,
      $p in //produit,
      $o in //commande[@cli-id=$c/@id and @prod-id=$p/@id]
return <res>{data($c/nom)}, {data($c/prenom)} :
        {data($o/quantite)} x {data($p/nom)}</res>
```

Quizz

N° de la question : 510

Quelles clauses de requêtes FLWOR sont également exprimables en XPath ?

A. For
B. Where
C. Order by
D. Group by

Répondre sur http://toreply.univ-lille1.fr

Question n° 510

Plan

- ✓ Introduction
- ✓ Documents XM
- ✓ Langage XQue
  - ✓ XPath
  - ✓ Requêtes FLWOR
  - ✓ Requêtes complexe

Sondage express

N° de la question : 735

Que pensez-vous de ce cours ?

Répondre sur http://toreply.univ-lille1.fr

Question n° 735