Bases de données avancées

Master 1 Informatique

2024-2025

Jérôme Darmont

https://eric.univ-lyon2.fr/jdarmont/

Actualité du cours

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

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

https://social.sciences.re/@darmont

#m1bda

Objectifs du cours

- SQL : langage de requêtes (bases de données relationnelles)
  - Standard
  - Optimiseurs de requêtes
  - Non procédural
  - Données structurées

- Programmation nécessaire pour :
  - Tâches complexes
  - Interfaces utilisateurs

Langage PL/pgSQL

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

requêtes

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)

CURSEURS

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)
- 2024 : v16.3

Michael Stonebraker

fr.wikipedia.org

Caractéristiques de PostgreSQL

- Types structurés (enregistrements, tableaux) possibles dans les tables (relationnel étendu)
- 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·e.

Répondre sur https://toreply.univ-lille.fr

Question n° 231

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
```

[ ] : clause optionnelle

(idem dans tout le document)

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 (      -- Définition
    num NUMERIC,
    nom VARCHAR
);  -- à définir hors de PL/pgSQL (commande SQL)

unEmploye tEmploye;            -- Déclaration
```

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;  -- C'est moche, mais
END LOOP;                 -- pas le choix...
```

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;
-- Eh oui, pas besoin 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

Répondre sur https://toreply.univ-lille.fr

Question n° 2

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();              -- La forme du résultat peut différer
SELECT * FROM test();       -- en fonction des clients PostgreSQL.
```

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);
-- ou
SELECT * FROM calculs(4, 5);
```

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

SELECT factorielle(10);
```

Sondage express

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

Répondre sur https://toreply.univ-lille.fr

Question n° 484

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 $$
	DECLARE
		e tEmploye;
	BEGIN
		FOR e IN SELECT * FROM emp LOOP
			e.nom := LOWER(e.nom );
			RETURN NEXT e;
		END LOOP;
		RETURN;
	END
$$ LANGUAGE plpgsql;

SELECT listeEmp();
```

De très loin
le plus courant !

Curseur explicite (lié)

```
-- Parcours ad-hoc du curseur   (même résultat que listeEmp → utiliser listeEmp !!)
CREATE OR REPLACE FUNCTION listeEmp2() RETURNS SETOF tEmploye  AS $$
        DECLARE
                cursEmp CURSOR FOR SELECT * FROM emp;
                e tEmploye;
        BEGIN
                OPEN cursEmp;
                FETCH cursEmp INTO e;           -- Lecture du 1er n-uplet
                WHILE FOUND LOOP
                        e.nom := LOWER(e.nom);
                        RETURN NEXT e;
                        FETCH cursEmp INTO e;   -- Lecture du n-uplet suivant
                END LOOP;
                CLOSE cursEmp;
                RETURN;
        END
$$ 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 $$
	DECLARE -- Pas de changement
	BEGIN
		OPEN cursEmp;
		FETCH cursEmp INTO e;
		WHILE FOUND LOOP
			e.nom := LOWER(e.nom);
			RETURN NEXT e;
			MOVE FORWARD pas FROM cursEmp; -- MOVE cursEmp; pour un seul décalage
			FETCH cursEmp INTO e;
		END LOOP;
		CLOSE cursEmp;
		RETURN;
	END
$$ LANGUAGE plpgsql;

SELECT * FROM listeEmp3(2);
```

Curseur paramétré (implicite)

```
CREATE OR REPLACE FUNCTION listeEmp5(salPlancher DECIMAL) RETURNS SETOF tEmploye AS $$
	DECLARE
		e tEmploye;
	BEGIN
		FOR e IN SELECT * FROM emp WHERE sal >= salPlancher LOOP
			e.nom := LOWER(e.nom);
			RETURN NEXT e;
		END LOOP;
		RETURN;
	END
$$ LANGUAGE plpgsql;

SELECT listeEmp5(2000);
```

Quizz

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 https://toreply.univ-lille.fr
Question n° 262

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 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 testErreur2(99);
```

Traitement par défaut

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

Fonctionnement des déclencheurs

BEFORE	AFTER
Requête sur table T ↓ Exécution du déclencheur ok ↓ ko ↓ BD ⊗	Requête sur table T ↓ BD ↓ Exécution du déclencheur

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;
```

| : ou

Variables systèmes des déclencheurs

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

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

  NEW.NumCli prend la valeur 1 dans un déclencheur sur client.
  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

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 https://toreply.univ-lille.fr

Question n° 191

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 (ex. changement de salaire)
    ⇒ 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

- Note :
  - Requête paramétrée : valeurs des attributs de la base de données
    → requête statique
  - Si l'on veut paramétrer des objets (tables, vues, nom d'attributs…)
    → requête 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)

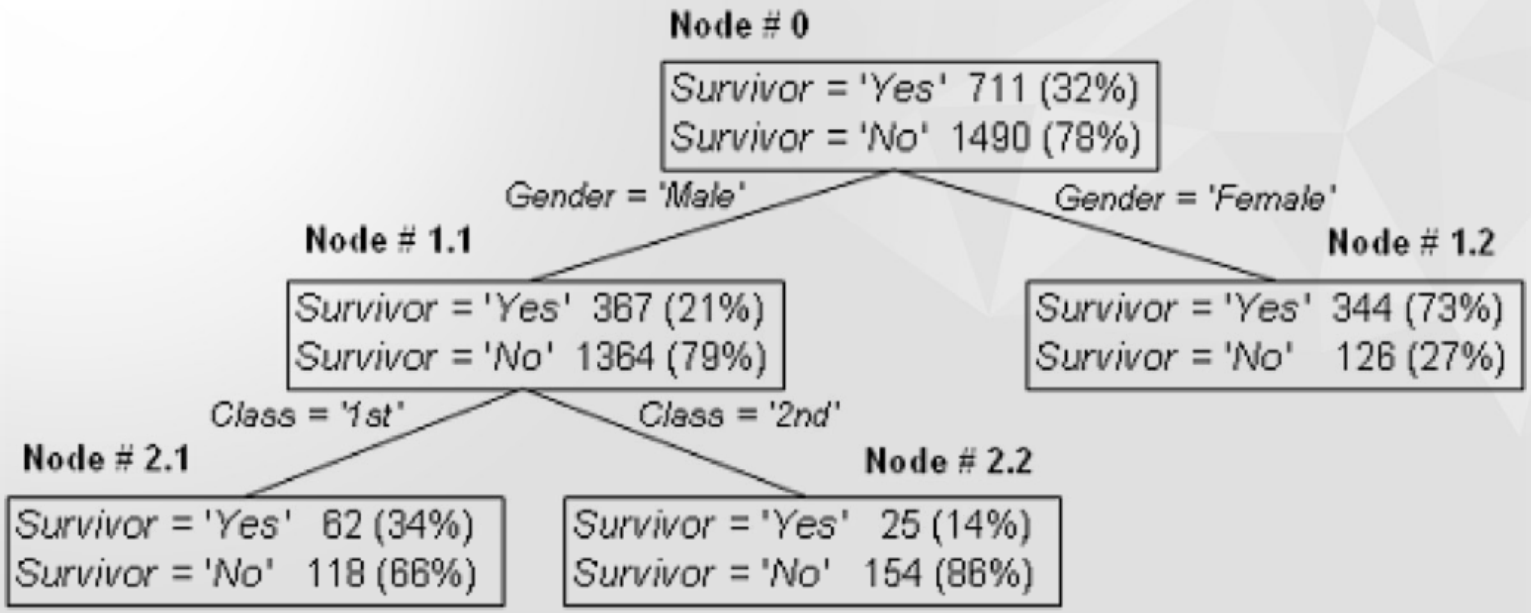
```
-- Exemple 2 : on utilise un SETOF VARCHAR (c'est moche, mais ça fonctionne...)
CREATE FUNCTION contenuTable(nomTable VARCHAR, nomID VARCHAR)
RETURNS SETOF VARCHAR AS $$
	DECLARE
		dynCurs REFCURSOR;
		id VARCHAR;
	BEGIN
		OPEN dynCurs FOR EXECUTE 'SELECT ' || nomID || ' FROM ' || nomTable;
		FETCH dynCurs INTO id;
		WHILE FOUND LOOP
			RETURN NEXT id;
			FETCH dynCurs INTO id;
		END LOOP;
		CLOSE dynCurs;
		RETURN;
	END
$$ LANGUAGE plpgsql;

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

Exemple issu de la recherche (1/3)

F. Bentayeb, J. Darmont, C. Favre, C. Udréa, "Efficient On-Line Mining of Large Databases", International Journal of Business Information Systems, Vol. 2, No. 3, 2007, 328-350.

- Problématique (à l'époque !)
  Contrainte de mémoire vive pour le *data mining* (arbres de décision)
- Hypothèse
  Intégration du *data mining* dans un SGBD
- Moyen
  Vues relationnelles
- Résultat
  Utiliser un SGBD est plus lent,
  mais permet de s'affranchir des problèmes de mémoire

Exemple issu de la recherche (2/3)



Rule #1: if *Gender* = '*Male*' and *Class* = '*1st*' then p(*Survivor* = '*Yes*') = 34%
Rule #2: if *Gender* = '*Male*' and *Class* = '*2nd*' then p(*Survivor* = '*No*') = 86%
Rule #3: if *Gender* = '*Female*' then p(*Survivor* = '*Yes*') = 73%

Exemple issu de la recherche (3/3)

Node #0: CREATE VIEW v0 AS SELECT Age, Gender, Class, Survivor FROM TITANIC;

Node #1.1: CREATE VIEW v11 AS SELECT Age, Class, Survivor FROM v0 WHERE Gender = 'Male';

Node #1.2: CREATE VIEW v12 AS SELECT Age, Class, Survivor FROM v0 WHERE Gender = 'Female';

Node #2.1: CREATE VIEW v21 AS SELECT Age, Survivor FROM v11 WHERE Class = '1st';

Node #2.2: CREATE VIEW v22 AS SELECT Age, Survivor FROM v11 WHERE Class = '2nd';

Quizz

Lesquelles de ces requêtes sont dynamiques ?

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

Répondre sur https://toreply.univ-lille.fr/

Question n° 828

Plan

- ✔ Introduction
- ✔ Bases du langa
- ✔ Curseurs
- ✔ Gestion des e eurs
- ✔ 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

Bases de données

Documents

Langages de requête

Moteurs de recherche

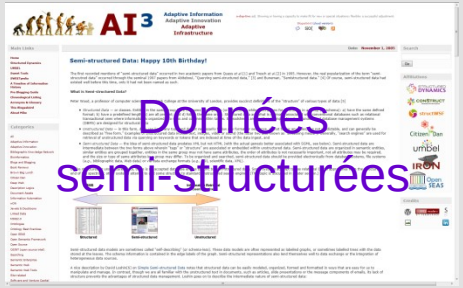
Années 1990

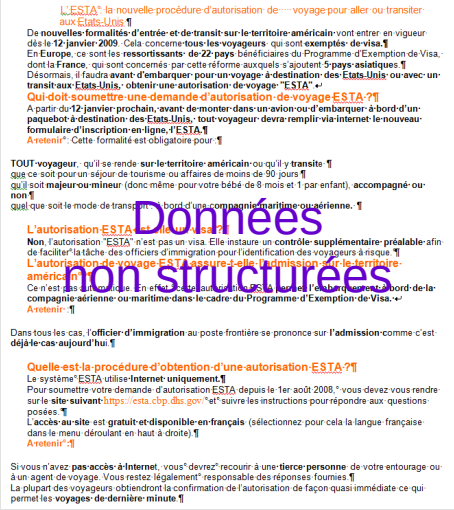
Données structurées

Données semi-structurées

Données non structurées

N° employé	Date embauche	Nom	Prénom	Adresse	Ville	Provinc	Code post	Date naissan
0090	12-janv-89	Charest	Eric	1075 rue Duluth	Montréal	QC	H4T 2L9	06-sept-49
0101	20-fév-89	Reys	Kimberly	899 ch. lakeshore	Donval	QC	H9P 3T2	15-déc-53
0150	05-mars-89	Langlois	Claude	308 rue Ste-Pierre	Montréal	QC	H2P 1A6	03-mars-68
0151	14-oct-89	Bédard	Julie	440 rue Talbot	Laval	QC	H7P 3G8	14-juil-45
0160	07-mars-89	Fortin	Jean-Paul	4380 rue Racicot	Montréal-Nord	QC	H2V 2P5	25-juil-59
0230	19-mars-89	Labrecque	Monique	892 ave. Victoria	Longueuil	QC	J3P 2T2	02-fév-37
0234	12-avr-89	Therrien	Jacques	320 rue Verchères	Greenfield Park	QC	J4V 2V6	02-fév-71
0367	21-avr-89	Proulx	Marcel	8085 boul. St-Laurent	Montréal	QC	H3V 2W3	06-mars-56
0380	12-mai-89	Labranche	Andrée	99 rue du Collège	Ville St-Laurent	QC	H4P 2M6	05-juil-64
0434	04-juin-89	Connell	Mark	1213 rue de Gaulle	St-Hubert	QC	J4T 2L9	28-fév-69
0600	10-juin-89	Reynolds	Alain	868 rue Ouellets	Outremont	QC	H3V 2V2	25-juil-67
0650	15-juin-89	Simard	Martin	2575 rue Victoria	St-Lambert	QC	J4L 6P7	03-avr-74
0728	15-juil-89	Bédard	Sébastien	3529 rue Rouvière	Ste-Thérèse	QC	J7P 3V5	14-janv-53
0839	14-août-89	Lebrun	Josée	694 de Montarville	St-Bruno	QC	J4L 3Y4	12-sept-71
1142	10-oct-89	Faucher	Martin	45 rue Handel	Candiac	QC	J2Y 2M7	31-oct-54
1151	14-oct-89	Bédard	Julie	621 rue Cannard	Laval	QC	H7P 3G7	02-juil-48
1241	07-janv-90	Simard	[illegible]	[illegible]	[illegible]	QC	J3V 5P9	25-nov-50
1260	01-fév-90	Morin	[illegible]	[illegible]	[illegible]	QC	H3W 2X3	06-fév-58
1341	04-mars-90	Verret	[illegible]	[illegible]	[illegible]	QC	J4P 1M2	13-mai-65
1460	05-nov-90	Quinn	Francis	4878 Côte-des-Neiges	Montréal	QC	H3Y 2P2	28-août-55
1473	07-nov-90	Vaudrin	Jean-Claude	2689 rue Fleury	Montréal	QC	H4M 1C5	12-déc-62
1481	01-déc-90	Wasfy	Maryse	383 de LaSalle	Boucherville	QC	J4Z 1T7	07-sept-68
1543	09-janv-91	[illegible]	[illegible]	[illegible]	Montréal	QC	H2T 2V6	30-mai-66
1560	08-juil-91	[illegible]	[illegible]	[illegible]	[illegible]	QC	H7P 7P8	03-juin-55
1571	14-août-91	[illegible]	[illegible]	[illegible]	[illegible]	QC	G0X 2X0	17-oct-62
2300	16-août-91	[illegible]	[illegible]	[illegible]	[illegible]	QC	J4P 1X3	26-sept-63
2312	04-nov-91	Lebrun	Carmen	1341 rue Robert	Montréal	QC	H3L 5M3	03-août-70
2315	04-nov-91	Couillard	Sophie	2689 boul. Décarie	Ville St-Laurent	QC	H4Y 2Z2	28-juil-69
2320	04-nov-91	Hamel	Nathalie	365 ave. Hutchison	Outremont	QC	H2V 2T4	15-mars-61
2321	01-déc-91	Milot	Bernard	8967 rue Viger	Montréal	QC	H3W 2M5	10-mai-73
2322	01-déc-91	Arel	Gilbert	436 rue Dawson	Montréal	QC	H3L 6L9	19-oct-67
2340	01-déc-91	Ouellet	Patricia	7228 boul. St-Joseph	Montréal	QC	H2H 3P6	13-avr-51
2341	05-janv-92	Archambault	Manon	673 Terr. Beaupré	Montréal	QC	H4G 2T1	31-déc-62
3389	04-fév-92	Brodeur	Guy	2575 Tisserand	Brossard	QC	J4P 2V3	26-juin-60
3490	04-mars-92	Lagacé	Mireille	592 St-André	Montréal	QC	H2T 5X2	26-fév-73
3500	04-mars-92	Paquette	Pierre	3687 Christophe-Colomb	Montréal	QC	H3V 5M1	09-nov-72
3501	04-sept-92	Trottier	Laurent	1955 rue Villeneuve	Longueuil	QC	J3P 6Y3	11-janv-72
3550	15-nov-92	Tremblay	Alexandra	524 ave. Dumont	Boucherville	QC	J4Z 8M2	14-fév-69
3554	15-nov-92	Voisard	François	2602 rue Perron	Montréal	QC	H4T 3M9	30-mars-66
3560	21-nov-92	Weber	Hans	52 rue Ste-Marie	Verchères	QC	J5G 4X2	20-sept-72
3561	21-nov-92	Gagnon	Eric	2235 rue Bréheuf	Montréal	QC	H3P 2J5	06-août-46
3580	09-déc-92	Demers	Marie-Claire	1275 rue Van Horne	Outremont	QC	H2V 6P5	05-nov-58
3800	08-déc-92	Robitaille	Céline	7929 Édouard-Montpetit	Montréal	QC	H3V 0L3	29-juin-74





L'ESTA : la nouvelle procédure d'autorisation de voyage pour aller ou transiter aux États-Unis

De nouvelles formalités d'entrée et de transit sur le territoire américain vont entrer en vigueur dès le 12 janvier 2009. Cela concerne tous les voyageurs qui sont exemptés de visa.
En Europe, ce sont les ressortissants de 22 pays bénéficiaires du Programme d'Exemption de Visa, dont la France, qui sont concernés par cette réforme auxquels s'ajoutent 5 pays asiatiques.
Désormais, il faudra avant d'embarquer pour un voyage à destination des États-Unis ou avec un transit aux États-Unis, obtenir une autorisation de voyage "ESTA".

Qui doit soumettre une demande d'autorisation de voyage ESTA ?

A partir du 12 janvier prochain, avant de monter dans un avion ou d'embarquer à bord d'un paquebot à destination des États-Unis, tout voyageur devra remplir via internet le nouveau formulaire d'inscription en ligne, l'ESTA.
A retenir : Cette formalité est obligatoire pour

TOUT voyageur, qu'il se rende sur le territoire américain ou qu'il y transite
que ce soit pour un séjour de tourisme ou affaires de moins de 90 jours
qu'il soit majeur ou mineur (donc même pour votre bébé de 8 mois et 1 par enfant), accompagné ou non
quel que soit le mode de transport : à bord d'une compagnie maritime ou aérienne.

L'autorisation ESTA est-elle un visa ?

Non, l'autorisation "ESTA" n'est pas un visa. Elle instaure un contrôle supplémentaire préalable afin de faciliter la tâche des officiers d'immigration pour l'identification des voyageurs à risque.

L'autorisation de voyage ESTA assure-t-elle l'admission sur le territoire américain ?

Ce n'est pas automatique. En effet, c'est au moment où vous vous trouvez à bord de la compagnie aérienne ou maritime dans le cadre du Programme d'Exemption de Visa.
A retenir :

Dans tous les cas, l'officier d'immigration au poste frontière se prononce sur l'admission comme c'est déjà le cas aujourd'hui.

Quelle est la procédure d'obtention d'une autorisation ESTA ?

Le système ESTA utilise Internet uniquement.
Pour soumettre votre demande d'autorisation ESTA depuis le 1er août 2008, vous devez vous rendre sur le site suivant : https://esta.cbp.dhs.gov/ et suivre les instructions pour répondre aux questions posées.
L'accès au site est gratuit et disponible en français (sélectionnez pour cela la langue française dans le menu déroulant en haut à droite).
A retenir :

Si vous n'avez pas accès à Internet, vous devrez recourir à une tierce personne de votre entourage ou à un agent de voyage. Vous restez légalement responsable des réponses fournies.
La plupart des voyageurs obtiendront la confirmation de l'autorisation de façon quasi immédiate ce qui permet les voyages de dernière minute.

Données semi-structurées

- Données organisées en entités sémantiques
- Entités similaires : groupes
- 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

Structurées

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
  jerome.darmont@msh-lse.fr

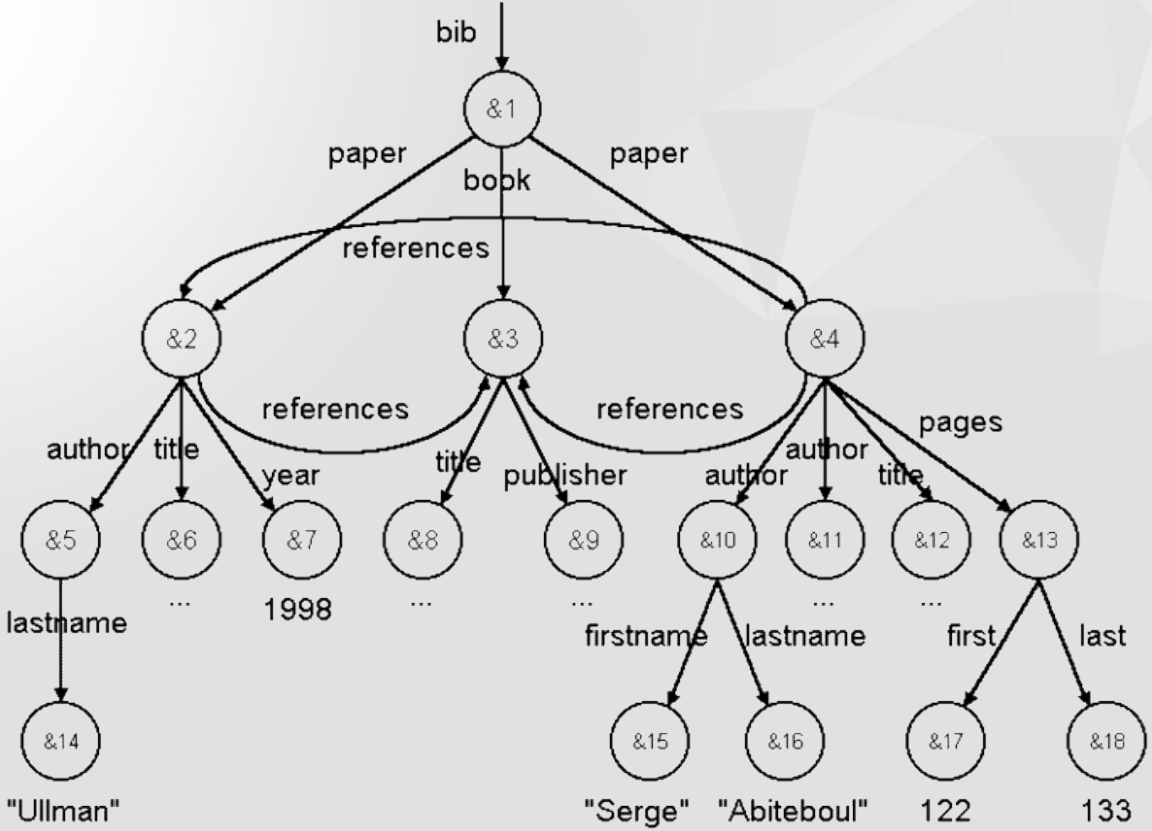
- Courriel — sabine.loudcher@univ-lyon2.fr
- Nom
  - Prénom — Loudcher
  - Nom de famille — Sabine

- Nom — Julien Velcin
- Affiliation — Université 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
  - Electronic Data Interchange (EDI) : domaine financier
  - Object Exchange Model (OEM) : modèle basé sur les graphes
  - SGML, HTML et XML
  - JSON, YAML…

Exemple de graphe OEM



Gestion de données semi-structurées

- Modélisation des données semi-structurées
  - Graphes (OEM) — Modèle conceptuel
  - DTD, XML-Schema — 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

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

Répondre sur https://toreply.univ-lille.fr/

Question n° 123

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 !

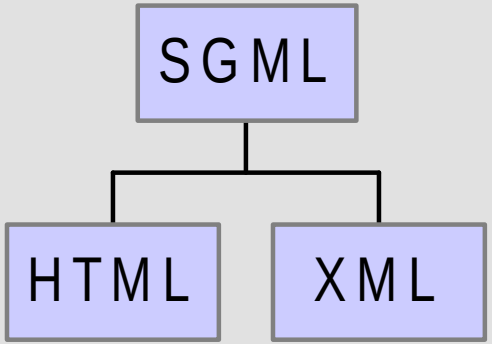
Plan

- ✔ 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) -->
<annuaireProfs> <!-- Élément racine -->
    <!-- Sous-éléments  -->
    <prof>
        <nom>Jérôme Darmont</nom>
        <courriel>jerome.darmont@univ-lyon2.fr</courriel>
        <cours>Bases de données avancées</cours>
        <cours>Programmation web backend</cours>
    </prof>
    <prof>
        <nom>Julien Velcin</nom>
        <courriel>julien.velcin@univ-lyon2.fr</courriel>
        <cours>Programmation orientée objet</cours>
    </prof>
    <!-- Etc. -->
</annuaireProfs> <!-- 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.
```
<annuaireProfs>
    <prof>
        <nom>
            <nom_famille>Zighed</nom_famille>
            <prenom>Abdelkader</prenom>
            <prenom>Djamel</prenom>
        </nom>
    </prof>
</annuaireProfs>
```

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">067</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 ?
  - <prof>
    <nom>Darmont</nom>
    </prof>
  - <prof nom="Darmont" />
- 4 principes pour décider
  - D'après Uche Ogbuji, Fourthought, Inc.

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

Répondre sur https://toreply.univ-lille.fr/

Question n° 722

Plan

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

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 (standard de fait)
- Soutenu par les éditeurs de SGBD (Oracle, Microsoft, IBM…)

Document XML exemple (1/2)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- le fichier s'appelle films.xml -->

<catVOD>

    <film sigJeune="-12">
        <titre>Blade runner</titre>
        <realisateur>Ridley Scott</realisateur>
        <annee>1982</annee>
        <langue>Anglais</langue>
        <prix>4.79</prix>
    </film>

    <film>
        <titre>La grande vadrouille</titre>
        <realisateur>Gérard Oury</realisateur>
        <annee>1966</annee>
        <duree>122</duree>
        <langue>Français</langue>
        <prix>9.82</prix>
    </film>                                    <!-- (…) -->
```

Document XML exemple (2/2)

```
<film sigJeune="-10">
        <titre>Le fabuleux destin d'Amélie Poulain</titre>
        <realisateur>Jean-Pierre Jeunet</realisateur>
        <annee>2001</annee>
        <duree>120</duree>
        <langue>Français</langue>
        <prix>4.99</prix>
</film>

<film sigJeune="-12">
        <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>9.82</prix>
</film>

</catVOD>
```

Plan

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

Expressions de chemins (1/3)

- Document XML entier
  doc("films.xml")/catVOD
  Résultat
  Tout le document

Par l'exemple

- Un élément donné
  doc("films.xml")/catVOD/film
  doc("films.xml")/catVOD/film/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("films.xml")/catVOD/film/data(@sigJeune)

  Résultat

  -12 -10 -12

- Un élément donné quel que soit son niveau hiérarchique

  /catVOD//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

  //film/*

  Résultat

  <titre>Blade runner</titre>
  <realisateur>Ridley Scott</realisateur>
  <annee>1982</annee>
  <duree>117</duree>
  <langue>English</langue>
  <prix>4.79</prix>
  <titre>La grande vadrouille</titre>
  <realisateur>Gérard Oury</realisateur>
  <annee>1966</annee>
  <duree>122</duree>
  <langue>French</langue>
  <prix>9.82</prix>

  Etc.

Prédicats XPath (1/2)

- i^e, dernier, i premiers/derniers éléments

  //film[1]
  //film[last()]
  //film[position() < 3]/titre

  Résultat
  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>

- Éléments possédant un sous-élément ou attribut donné

  //film[duree]/titre

  Résultat
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>

  //film[@sigJeune]

Prédicats XPath (2/2)

- Condition sur un élément ou un attribut

  //film[prix < 15]

  //film[@signJeune = "-10" and prix < 5]/titre

  Résultat

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

- Combinaison de chemins

  //titre | //prix

  Résultat

  <titre>Blade runner</titre><prix>4.79</prix>

  <titre>La grande vadrouille</titre><prix>9.82</prix>

  <titre>Le fabuleux destin...</titre><prix>4.99</prix>

  <titre>The big Lebowski</titre> <prix>9.82</prix>

Fonctions XPath

- 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(), sort()…
- Fonctions contextuelles : last(), position()…
- Fonctions booléennes : not()…
- Fonctions d'agrégat : count(), sum(), avg(), max(), min()…

Quizz

Lesquels de ces chemins sont corrects ?

A) /catVOD/film/prix
B) doc("films.xml")/catVOD/film/prix
C) //prix
D) /catVOD//prix

Répondre sur https://toreply.univ-lille.fr/

Question n° 492

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),
     $y in (10, 20) (: Ceci est également un commentaire :)
return <x>{$x}</x> <y>{$y}</y>
```

Résultat

```
<x>1</x> <y>10</y>
<x>1</x> <y>20</y>
<x>2</x> <y>10</y>
<x>2</x> <y>20</y>
```

Clause For (3/3)

Exemple

```
for $x at $i in //film/titre
return <film id="{$i}">{data($x)}</film>
```

Résultat

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

Clause Let

- Clause Let : Assigner une ou plusieurs valeurs à une énumération (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 //film
where $x/prix > 5
return $x/titre
```

Exemple

```
for $x in //film
where $x/@sigJeune = "-12" and $x/prix < 10
return $x/titre
```

Clauses Order by et Return (1/2)

- Clause Order by : Trie le résultat

Exemple

```
for $x in //film
order by $x/titre
return $x/titre
```

Exemple

```
for $x in //film
order by $x/@sigJeune, $x/titre descending
return $x/titre
```

- Clause Return : Spécifie le résultat (un ou des fragments XML)

Clause Return (2/2)

- Expressions conditionnelles

Exemple

```
for $x in //film
return   if ($x/@sigJeune ="-10" or sigJeune = "-12")
         then <sigOK>{data($x/titre)}</sigOK>
         else <sigKO>{data($x/titre)}</sigKO>
```

Résultat

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

Fonctions XQuery (les mêmes que XPath !) (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(), sort()…
- Fonctions contextuelles : last(), position()…
- Fonctions booléennes : not()…
- Fonctions d'agrégat : count(), sum(), avg(), max(), min()…

Fonctions XQuery (2/2)

Exemple d’appel à une fonction

```
for $x in doc("films.xml")/catVOD/film/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 /catVOD/film
group by $z := $d/@sigJeune
return    <sigJeune value="{$z}">
              <prix_moyen>{avg($d/prix)}</prix_moyen>
          </sigJeune>
```

Regroupement multiple

```
for $d in /catVOD/film
group by $z := $d/@sigJeune, $a := $d/annee
return    <groupe sigJeune="{$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 K067</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 = $o/@cli-id
return     <nom>{data($c/nom)}</nom>
           <prenom>{data($c/prenom)}</prenom>
           <qte>data($o/quantite)}</qte>
```

Résultat

```
<nom>Loudcher</nom> <prenom>Sabine</prenom> <qte>3</qte>
<nom>Loudcher</nom> <prenom>Sabine</prenom> <qte>15</qte>
<nom>Bentayeb</nom> <prenom>Fadila</prenom>  <qte>7</res>
<nom>Bentayeb</nom> <prenom>Fadila</prenom>  <qte>10</qte>
<nom>Darmont</nom> <prenom>Jérôme</prenom> <qte>5</qte>
```

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 = $p/@id
return     <nom-cli>{data($c/nom)}</nom-cli>
           <prenom>{data($c/prenom)}</prenom>
           <qte>{data($o/quantite)}</qte>
            <nom-prod>{data($p/nom)}</nom-prod>
```

Résultat

```
<nom-cli>Loudcher</nom-cli> <prenom>Sabine</prenom>  <qte>3</qte>   <nom-prod>Ordinateur</nom-prod>
<nom-cli>Loudcher</nom-cli> <prenom>Sabine</prenom>  <qte>15</qte> <nom-prod>Moniteur</nom-prod>
<nom-cli>Bentayeb</nom-cli> <prenom>Fadila</prenom>   <qte>7</qte>   <nom-prod>Ordinateur</nom-prod>
<nom-cli>Bentayeb</nom-cli> <prenom>Fadila</prenom>   <qte>10</qte> <nom-prod>Imprimante</nom-prod>
<nom-cli>Darmont</nom-cli>  <prenom>Jérôme</prenom> <qte>5</qte>   <nom-prod>Imprimante</nom-prod>
```

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 <nom>{data($c/nom)}</nom> <prenom>{data($c/prenom)}</prenom>
          <qte>{data($o/quantite)}</qte>
```

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

Jointures de documents XML

FLWOR
=
lisibilité

XPath
=
concision

Quizz

Quelles clauses de requêtes FLWOR sont également exprimables en XPath ?

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

Répondre sur https://toreply.univ-lille.fr/

Question n° 434

Plan

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

Sondage express

Que pensez-vous de ce cours ?

Répondre sur https://toreply.univ-lille.fr

Question n° 407