Persistance des données :

Placentino

May 28, 2015

Contents

Evaluations du cours :

- Interrogation en Janvier 25%
- Examen en Mai/Juin 75%

Objectifs du cours :

- Pouvoir percevoir et comprendre les différents niveau de description des données.
- Maitriser du SQL-DML (DML = Data Manipulation Language, c'est la partie C.R.U.D.)
- Bonne connaissance du DDL (Data Definition Language).

Plan du cours :

1. Introduction
2. Modèle Relationnel
3. S.Q.L. (Non procédurale, le S.G.B.D. le traduit en procédural)
4. Embedded-SQL (Utiliser SQL pour interroger, manipuler des données persistantes)
5. Programmation sur serveur
6. Sécurité
    - Détails sécurité physique
    - Détails sécurité d'accès
    - Sécurité logique (les données soient bien coérantes)
        - Concurrence d'accès

17/09

Il existe plusieurs familles de S.G.B.D.[1][2]

- Relationnel
- hiérarchique
- réseau
- ...

- Systeme = Ensemble de programmes offrant des services, des facilités aux utilisateurs.
- Gestion =
  - Manipulations des données (manipulations = C.R.U.D.)
  - sécurisation des données(octroiement des droits d'accès, physique, cryptage, logique)
  - définir (définir la structure des données),
  - ...

Contrainte d'intégrité : une donnée non-intègre != une donnée incorrecte. C'est une règle que les données doivent respecter (Ex. : Le prix d'un article DOIT être positif). Notion qui existe dans le monde réel et qui ne provient pas explicitement du monde informatique. "C'est une règle que les données doivent suivre pour représenter un monde réel possible" ADT, 2014. Une base de donnée est coéréante quand toutes les contraintes d'intégrité sont respectées.

```
1. Introduction


    1.1. Etude du monde réel

        La redondance est génératrice d'incohérences ->
            On essayera toujours d'avoir une modélisation unique du monde réel.

        Modèle Conceptuel de Données, une représentation du réel qui semble nous suffire, co

    1.2. S.G.F.
```

[1] Système de Gestion de Base de Données
[2] D.B.M.S. (Data Base Managment System)

M.C.D. -> { de fichiers représentant çela } (M.P.D. Modèle Physique des Données).
Le M.P.D. DOIT représenter tout le M.C.D.
Le M.C.D. est l'agrégation des visions de chacun

Problèmes :
* Sécurité : Demander la liste des employées nous donne accès a toutes les i
* Réinterprétation des données.
* Evolution du M.C.D.
* Performance : elles se dégradent en fonction du temps (évolution du volume
* Contrainte d'Intégrité, écrite par le programmeur et redondant sur l'ensem
régies par ces contraintes. Cette contrainte prend sa source dans le monde r
alors les données sont cohérantes.
* D.D. : Description des données est éclatée entre la document du S.G.F. et

1.3. S.G.B.D.

G.Gardarin, BD objets et relationnelles.

- Disposer de l'indépendance logique :
  Une seule réinterprétation pour l'ensemble des programmes liés à UNE vision des
  Si les données d'une vision change, il suffit de changer la vision.
  Ex. : le cube.
  Apprendre au SGBD a travailler avec une vision des données.

- Disposer de l'indépendance physique :
  Pouvoir modifier la structure des données, cette modification sera prise en char
  Les performances vont donc évoluer (+ ou - si je fais de la merde).

- Absence de redondance (esperée)
  On espere que tout fait du monde réel soit exprimé une fois au maximum.

- Intégrité des données
  = cohérence des données = Data consistancy
  Faire apprendre les contraintes d'intégrités par le SGBD, on ne l'écrit donc qu'

- Partagabilité des données
  Ex : on peut utiliser un ATM même si qqun retire déjà de l'argent autre part.

- Sécurité des données :
  sécurité physique : Se mettre à l'abri des conséquences de problème physique (cr
  " d'accès : Seul les personnes autorisées ont le droit de faire ..
  " logique : gèrer la concurrence d'accès en disposant d'outil nous permettant

- Administration centralisée des données
- Manipulation des données par des non-informaticiens

1.4 ANSI- X3 - SPARC

Les schémas de définition de données.

Les descriptions des données doivent être fournies en 3 type de schémas distincts :

- Schéma central = schéma conceptuel
  le plus proche du monde réel pour celui qui s'occupe de BD
  = Ensemble de tables à décrire (traduction du M.C.D.)
- Schéma externe = Constructions réalisées sur le schéma conceptuel (ce sont les
  PROMET L INDEPENDANCE LOGIQUE
- Schéma (unique) interne = Comment sont stockées les différents élement et quel

ANALYSTE	BDISTE
M.C.D.	
M.L.D.	SCHEMA CONCEPTUEL (trad. du M.C.D.)
M.P.D.	SCHEMA CONCEPTUEL + SCHEMA INTERNE

L'ensemble des données de la description des données = METADONNES, elles sont stocké

SYSTABLES	Tno	Tnom
	423	Etudiant

SYSCOLUMNS	Cno	Cnom	Ctable	Cordre	Ctype
	1737	EtuNo	423	1	int
.....					

2. Modlèle relationnel.

1. Le modèle hiérarchique

Le schéma conceptuel est nécessairement un arbre.
Chaque element peut avoir de 0 à n enfants, et ainsi de suite.
Default majeur : lors de la conception du schéma conceptuel, on privilège déjà certa

2. Modèle réseau

Même concept mais on est plus limité aux arbres mais on peut utiliser des graphes

3. Modèle relationnel
4. Modèle Orienté Objet (on vera en 3ème que dans certains environnement c'est mega perf
5. Modèle Relationnel Objet : le relationnel vie à l'interieur du R.O. car le relationne
6. NoSQL (= Not Only SQL)

Les produits relationnels :

-MySQL
-PostgreSQL
-Oracle
-SQLite
-javaDB (- derby)
-Acces
-SQLserver
--DB2
- Ingres
- Rbase
- ..

3. Les concepts de base

Codd a défini les bases du modèle relationnel à partir de la théorie des ensembl
Il a exprimé des choses de manière non procédurale (A U B en procédurale s'expri

1ère notion : Domaine
est un ensemble (ex.: l'ensemble des entiers, des dates, des entiers pai
deux fois le même élément. L'ordre n'a pas de sens.

2eme : Domaines compatibles
lorsque nous pouvons comparer des éléments de chacun d'eux.

3eme : Relation
sous-ensemble d'un produit cartésien
ex : R1 = {(dupont, vrai), (dupuis, faux)}

```
R1 | Nom : D1 | Cadre : D2      : SCHEMA DE LA RELATION
   | DUPONT   | VRAI            : EXTENSION DE LA RELATION
   | DUPUIS   | FAUX
```

sémantique : quel est le fait ou les faits représenter par les données.

Relation	Table
tuple	ligne (row)
attribut	nom de colonne

==> Base de données relationelle : est une base de données dont le schema co
d'un ensemble de schema de relation.
ex. Dno : naturel
Dchar50 :
Dsec : { I, R, G}
Dan : {1, 2, 3}

Etudiant(etuNo : Dno, etuNom : Dchar50, etuSec : Dsec, etuAn : Dan)
Cette relation représente les élèves regulièrement inscrits à l'esi
pour lesquels etuno represente son numero, ...
Degré d'une relation : nombre d'attributs de la relation, (de colonn

```
            Deux relations sont compatibles : les deux relations doivent avoir l
            être compatibles deux à deux (1ere 1 ere, 2eme 2eme,..)

            Le modèle relationnel (pas sql) accepte plusieurs type d'absence de
            très problématique :
            comparaison avec NULL

    4. Algèbre relationnelle

    4.1. Intersection ( ^ )

        Soient R1, R2 2 relations compatibles
            R1 ^ R2
        SuitCoursC ^ SuitCoursBD    | etuNo     | etuNom
        ------------------------------------------------------
                        | 26        | Durant

    4.2. Union ( U )

        Soient R1, R2 2 relations compatibles
            R1 U R2 = R1 + R2 - R1 ^ R2

    4.3 difference ( - )

        soient R1, R2 deux relations compatibles
            R1 - R2 = R1 - (R1 ^ R2)
            les tuples présents en R1 mais pas dans R2.

    4.4 projection :

        Soit relation R(a, b, c, d, e)
        R[a] ne garde que les valeurs de la colonne a
        R[b] ..
        R[c] ..

    4.5 Selection :

        R(cond)
```

Exemple :

voir feuille annexe.

```
    4.6. Produit cartésien :
```

```
Soient R1(A1, ..., An)
et R2(B1,..., Bm)

R1 x R2 de degré n + m

jointure :
    interne : INNER JOIN
        JOIN (A, B; a1 = b3)
        traduit en : A x B (a1 = b3)

    exerne : OUTER JOIN
        -Droite (right)
        -Gauche (LEFT)
        -Complete (FULL)
```

Dependance fonctionnelle : (propriétés du monde réel, contraintes d'intégrité) R(A, B, C, .., N) A -> B

Commande(com, cli, date, pro, Q, pu)

com -> cli, date cli -> com, pro -> Q, pu cli, date -> com

soient R(A1, A2, .., An) une relation et X, Y des sous-ensembles de {A1, ..., An} on dit que X -> Y si pour tout extension r de R, pour tout t1, t2 appartenant à r on a R[X].t1 = R[X].t2 (projection de R selon X) => R[Y].t1 = R[Y].t2

Prouvez que telle dépendance est fausse en regardant une table

PROPRIETE DES DF :

```
Transitivité :

A -> B et B -> D        => A -> D

Reflexivite :

A -> A

Augmentation :

si A -> B alors A, C -> B

Additivité :

si A -> B et C-> D alors A, C -> B, D

Pseudotransitivité :
```

```
si A -> B et B, C -> D alors A, C -> D

    Fermeture transitive :
        Dependances fonctionnelles -> X dependences fonctionnelles
        (fermeture transitive) -> couverture minimale (plus petits nombre de
        dépendances fonctionnelles mais ayant la meme fermeture transitive)
```

Clé de relation : Sous ensemble minimal d'attributs qui déterminent tous les autres

```
Commande(com, cli, date, pro, Q, pu)

        com, pro -> cli, date, Q, pu
        cli, date, pro -> com, Q, pu

        => 2 clefs de relation;

clé : sous ensemble x d'attributs de R(A1, .., An):
        I.  X -> A1, ..., An
        2.  Il n'existe pas de y non inclut dans X : y -> A1, .., An

        Ex : com, pro, Q n'est pas une clé car si on retire Q on peut
        toujours déterminer tous les attributs.

clé candidate : clé qui n'a pas été choisie pour être primaire.


Exemple :

    VEH(chassis, plaque, marque, ..)


Exercice 1.
    a) faux
    b) vrai
    c) faux
    d) faux plus petit ou egal
```

Les formes normales :

```
DF :
    etuId -> etuNom, etuNat, etuDom
    etuNat -> etuNatId
    etuId, insAnnac -> anet, catId, oriId
```

```
    oriId -> oriNom, oriTp
    crsId -> crsLib, crsNbH, crsType
    catId -> catLib

1ère FN (Forme Normale) :

    etuId, insAnnac, crsId

    On parle de 2 ^ n - 1 choses (7 là).
    => On est assuré maintenant qu'un eleve ne suit pas deux fois le même
    cours.(gain faible).
2ème FN :

    création de 7 tables :
    Etudiant(etuId, etuNom, etuNat, etuNatLib, etuDom) | etuId = cle CANDIDATE
    AnneeAcad(insAnnac) | insAnnac = cle CANDIDATE
    Cours(crsId, crsLib, crsNbH, crsType)
    Inscription(etuId, insannac, anet, oriId, oriNom, oriTpEm, catId,
    catLib)
    ASuivi(etuId, crsId)        [Table un peu compliqué..]
    EstOrganisé(insAnac, crsId)
    Suit(etuId, insAnac, crsId)

    + de redondances
    + gain de représentativité
        (représenter des choses sensé que je ne pouvais pas me représenter
        en première forme normale. Ex: ajouter un cours sans encore aucun
        étudiant qui le suis).
    - d'incohérences

    C.I.R. : Contraintes d'Intégrité Référencielles

        Inscription[etuId] C Etudiant[etuId]
        Inscription[insAnnac] C AnneeAcad[insAnnac]
        aSuivi[etuId] C Etudiant[etuId]
        aSuivi[crsId] C Cours[crsId]
        estOrganise[insAnnac] C AnneeAcad[insAnnac]
        estOrganisé[crsId] C Cours[crsId]
        Suit[etuId, insAnnac] C Inscription[etuId, insAnnac]
        Suit[insAnnac, crsId] C estOrganise[insAnnac, crsId]
        Suit[etuId, crsId] C ASuivi[etuId, crsId]

        => Apres les CIR on vient d'arriver en 2eme forme normale


        2eme FN SSI tout attribut n'appartenant pas à une clef ne dépend pas
```

```
        d'une partie d'une clef + 1FN


3FN :

    Etudiant(etuId, etuNom, etuNat, etuDom)
    Pays(paysId, paysLib)

    Inscription(etuId, insanac, anet, oriId, catId)
    Orientation(oriId, oriNom, oriTpEns)
    Categorie(catId, catLib)

    C.I.R. en plus :
        Etudiant[etuNat] C pays[paysId]
        Inscription[oriId] C Orientation[oriId]
        Inscription[catId] C Categorie[catId]


        3eme FN SSI 2FN + tout attribut n'appartenant pas à une clef ne
        dépendant pas d'un attribut non clef



R(A, B, C, D, E, F)

Si A, B = clef => 1FN

Si B -> C => pas 2FN
si D -> E => pas 3FN
```

/!! Interrogation : ajout de table(s) intermédiaire(s)

etrangere -> primaire etrangere C primaire

Chapitre 4 : S.Q.L.

4.1. intro Language normalisé (ISO SQL86-SQL92 et ANSI) et donc partiellement portable.

4.2. La structure de SQL

```
DML : DATA MANIPULATION LANGUAGE
    4 instructions :
        R = SELECT
        D = DELETE
        U = UPDATE
        C = INSERT
```

```
DDL : DATA DEFINITION LANGUAGE
    CREATE : Créer une table ou ....
    DROP {table, view, cluster} : Supprimer
    ALTER {nomObjet} : Modifier définition d'objet

DCL : DATA CONTROL LANGUAGE
    REVOKE
    GRANT

Quelques règles syntaxiques :
    délimiteur d'instruction :
        ; \
        espaceS = espace
        commentaires :
            -- (= //)
            /* */ (= /* */)
    délimiteurs de chaines :
        ' doublé si utile dans la chaine
        " = délimiteur d'identifiant CASE SENSITIF, sinon tout en MAJ
    //!\\ INTERROGATION "
```

4.3. DML de SQL 4.3.1. SELECT Peut être imbriqué dans un EXIST()

```
4.3.2. UPDATE
    Mettre à jour, modifier, des tuples déjà existant.
    UPDATE nomTable
        SET att1 = expr1
            [,att2 = expr2]
            [, ...]
        [WHERE cond]

    Ex. :
        Multiplier par deux le salaire de Dupont :

        UPDATE employe
            SET empSal = empSal * 2
            WHERE empNo = ..

    Ex. :

        UPDATE employe mgr
            SET empSal = 2 * (
                                SELECT MAX(empSal)
                                    FROM employe
```

```
                                    INNER JOIN Departement
                                    ON empDpt = dptNo
                                    WHERE mgr.empNo = dptMgr
                            )
        WHERE empNo IN
                ( SELECT dptMgr FROM Departement ) ;
```

/!! Toutes les instructions en DML disposent de “l’intégrité en lecture”. => Thread safe, la lecture se fait sur une IMAGE figée. Soit, l’exemple précédent cohérant même si un employé est un manager. Le SGBD gère ça tout seul.

```
    + Atomicité

Retourne un "SQLcode" qui vaut :
    --> 0   "OK"
    --> >0  "OK mais j'ai un truc à dire Farit" /WARNING\
                Ex : update avec un where il y a aucune ligne à modifier.
    --> <0  "ERREUR"    Syntaxique "selct", Sémantique "select machin"
                        n'est pas un attribut de la table, de droit :
                        modifier une table de ADT.


4.3.3. DELETE
    DELETE FROM table
        [WHERE cond]

    Ex. :

        DELETE FROM Employe
            where empNo = ..

4.3.4. INSERT
        a)
            INSERT INTO nomTable[(attr1, attr2, attr3, ..)
                VALUES(.., .., .., ...)

        b)
            INSERT INTO nomTable[(attr1, attr2, attr3, ..)
                SELECT ...;

                => créer de la redondance, prendre des données quelque part
                et les recopier quelque part.
```

4.4 D.C.L.

```
Transaction = Sous ensemble minimum d'instructions qui fait passer une base
de données d'un état cohérant à un état cohérant.

    ACID :
        A : Atomicité = indivisible.
        C : Consistency = cohérance.
        I : Isolation = Si les transactions se passent en même temps, elle
        ne s'influent pas mutuellement.
        D : Durability = quand la transaction est terminée, les modifs sont
        répertoriées dans la BDD.


    begin Transaction
        - try {
        |
        |
        |
        |
        | VALIDATION DE LA TRANSACTION
```

catch (ERREUR) | | ANNULATION DE LA TRANSACTION | - end Transaction

```
COMMIT
ROLLBACK
    Ne fonctionnent que sur instruction du DML. DDL genere obligatoirement
    un commit par lui même.
```

4.5 Le DDL définitions des schémas de données

4.5.1. Le schéma conceptuel

```
a) Définition minimale d'une table

    CREATE TABLE Departement(
        dptNo CHAR(3),
        dptLib VARCHAR(30),     // varchar = character varying
        dptAdm char(3)
        );


b) Les types de données

    Les chaines
```

- CHAR[ACTER][(n)]
- VARCHAR(n)

Les chaines de bits

- BIT[(n)]
- BITVAR(n)

Les nombres

- NUMBER(precision, scale) // (nombre de chiffres, nombre apres la virg)
- DECIMAL(precision, scale)
- FLOAT, REAL, DOUBLE PRECISION
- INT ou INTEGER

Moments

- DATE
- TIME
- TIMESTAMP

LOB (Large Object) (stockage brut de gros fichiers)

- CLOB // Characters LOB ex. : file.doc
- BLOB // Binary LOB ex. : file.mp3

c)Caractère obligatoire d'un attribut

```
CREATE TABLE Departement(
    dptNo CHAR(3) NOT NULL,
    dptLib VARCHAR(30) NOT NULL,        // varchar = character varying
    dptMgr char(3) NOT NULL,
    dptAdm char(3) [NULL/NOT NULL] [DEFAULT {lit}]
    );
```

d) Valeur par défaut

```
INSERT INTO Departement(dptNo, dptLib, dptMgr)
    VALUES('A21', 'dfsfds', 'B33');
    => dptAdm sera par defaut à NULL si DEFAULT n est pas présent.
```

Contraintes :

```
CREATE TABLE Departement(
```

```
dptNo CHAR(3) NOT NULL,
dptLib VARCHAR(30) NOT NULL,        // varchar = character varying
dptMgr char(3) NOT NULL,
dptAdm char(3)
CONSTRAINT DepartementPK PRIMARY KEY(dptNo)
);
```