Langage SQL (1)

Sébastien Limet Denys Duchier

IUT Orléans

4 septembre 2007

Notions de base

- qu'est-ce qu'une base de données ?
- SGBD
- différents type de bases de données
- quelques systèmes existants

Définition d'une BD

Une BD est :

- un ensemble structuré
- de données persistantes
- représentant une réalité extérieure au système
- partagé par plusieurs applications
- d'une même entreprise

SGBD : objectifs

Un système pour gérer :

- les données
- l'accès aux données

SGBD : objectifs “données”

- éviter les redondances de données
  - *occurrences multiples d’un même étudiant*
- éliminer les incohérences
  - *un étudiant non-inscrit suivant un cours d’info*
- gérer des types de données très variés
  - *numériques, booléens, texte, image, sons*

SGBD : objectifs “accès”

- standardisation de l’accès aux données
  - *langage de requêtes*
- sécurité
  - *utilisateurs, droits, rôles, authentification*
  - *cryptage des données*
- robustesse
  - *disponibilité des données, résistance aux pannes*
  - *performances*
- mode client/serveur
  - *gestion des accès concurrents*
- répartition des données
  - *sur plusieurs machines, plusieurs sites*

SGBD : objectifs “modèle”

Surtout : permettre une description logicielle unique, indépendante des caractéristiques d’implantation

La base de données est vue par ses utilisateurs comme un modèle logique des données (MLD) complètement indépendant de son implantation physique.

Différents types de BD

- modèle navigationel
- modèle relationel
- modèle orienté-objet

Modèle navigationel

S'appuie sur les notions de pointeur et de chemin pour naviguer dans la base des enregistrements.

- bases hiérarchiques :
  - *premiers SGBD*
  - *organisation des entités selon un arbre*
  - *liaisons entre entités par pointeurs*
- bases réseaux :
  - *les plus rapides*
  - *schéma plus ouvert que hiérarchique*

Modèle relationel

S'appuie sur les notions de relations et de langage déclaratif pour la formulation de requètes.

- bases relationelles :
  - *les plus utilisées*
  - *représentation en tables*
  - *algèbre relationelle*
  - *langage déclaratif d'interrogation (SQL)*
- bases déductives :
  - *représentation en tables/prédicats*
  - *langage d'interrogation basé sur le calcul des prédicats et la logique du 1er ordre*

Modèle orienté-objet

S'appuie sur les notions d'objet et de champ, mais associe parfois à cette approche navigationelle un langage d'interrogation déclaratif.

- moins utilisée
- les données sont représentées en terme de classe ; un champ contient un objet
- chaque donnée est donc active
- l'héritage est utilisé comme mécanisme de factorisation de la connaissance
- introduction des traitements directement dans la BD

Systèmes existants

Tous sont relationnels ; certains introduisent un peu d'objet.

- Oracle
- DB2
- Postgres
- Informix
- SysBase
- SQL server
- Access
- MySQL

Niveaux de langages BD

- langage de définition de données (LDD)
  - *définition, destruction de tables, utilisateurs, etc. . .*
- langage de manipulation de données (LMD)
  - *insertion, mise à jour, effacement de données*
- langage de requêtes
  - *consultation de la base de données*

SQL permet toutes ces opérations !

Vocabulaire

En SQL :

- une table est l'équivalent d'un fichier
- une colonne est l'équivalent d'un champ
- une ligne est l'équivalent d'un enregistrement

Préambule

une colonne `C` d'une table `T` se désigne par :

`T.C`

quand il n'y a pas d'ambiguité, on peut abréger en :

`C`

l'ensemble des colonnes d'une table `T` se désigne par :

`T.*`

Conventions

- ⟨Chose⟩ désigne un élément syntaxique nommé “chose” qui sera défini par une ou plusieurs règles syntaxiques.
- **`MotClé`** désigne un mot clé ou une ponctuation du langage
- on écrira l’ensemble des règles pour un élément syntaxique de la manière suivante :

$$\langle \text{Chose} \rangle \quad ::= \quad \langle \text{règle1} \rangle \mid \langle \text{règle2} \rangle \mid \cdots \mid \langle \text{règle} \rangle$$

Définition des données

- changer son mot de passe :

```
alter user <USER> identified by <PASSWD>;
```

- créer une table :

```
create table <IDENT> (<COLS>);
```

Exemple :

```
create table client (
    numcli  Number(4),
    nom     Varchar2(20),
    prenom  Varchar2(20),
    adresse Varchar2(20));
```

- supprimer une table :

```
drop table <IDENT>;
```

Types de données courants

- **`Char`**(taille) : chaîne de caractères de taille fixe
- **`Varchar`**(taille) : chaîne de caractères de taille variable
- **`Varchar2`**(taille)
- **`Number`**(taille,décimales)
- **`Date`**
- **`Blob`** : jusqu'à 8 téraoctets (10^{12} ou 2^{40}) de données binaires

Constantes et conversions

- constantes :
  - **Number** : *19.4*
  - **Char**, **Varchar** : *'Bonjour'*
- conversions :
  - `to_date('19-12-2004', 'dd-mm-yyyy')`
  - `to_date(date, 'dd-mm-yyyy')`
  - `to_number('123')`
  - `to_char(23.5)`

Manipulation des données

- insertion d'une ligne :

```
insert into <TAB> values (<VALS>);
```

- exemple :

```
insert into client values
    (124, 'Dupont', 'Jean', 'Orleans');
```

Manipulation des données

- suppression d'une ligne :

  ```
  delete from <TAB> where <CONDITION>;
  ```

  un "delete" sans clause "where" provoque l'effacement de toutes les données de la table

- exemples :

  ```
  delete from client where
      adresse='Orleans';

  delete from client;
  ```

Manipulation des données

- mise à jour de lignes :

```
update <TAB>
    set <COL> = <VAL>,
    ...
    where <CONDITION>;
```

- exemples :

```
update client
    set   prenom='Paul'
    where numcli=123;

update client
    set  adresse='ORLEANS';
```

Manipulation des données

Les parties définition et manipulation des données sont beaucoup plus riches :

- contraintes d'intégrité
- clés
- manipulation de vues
- etc. . .

Client/Serveur

Attention : les insertions, mises à jour et suppressions de lignes sont différées.

- validation des commandes : **`commit`** ;
- annulation des dernières commandes : **`rollback`** ;

à la déconnexion : **`commit`** automatique !

Requêtes SQL

- Structured Query Language
- principal langage de requêtes des BD relationelles
- basé sur l'algèbre relationelle

Principales opérations

- Projection : restriction sur les colonnes
- Sélection : restriction sur les lignes
- Produit cartésien : association de toutes les lignes d'une table à toutes les lignes de l'autre
- Jointure : association de certaines lignes d'une table avec certaines lignes d'une autre (reconstitution de l'information)

Afficher le contenu d'une table

```
select * from Adherent;
```

Projection

```
select nom, prenom from Adherent;
```

Sélection

```
select * from Adherent
    where nom='Dupont';
```

Produit cartésien

```
select nom, prenom, intitule
    from Adherent, Participe;
```

Jointure

```
select nom, prenom, intitule
    from Adherent, Participe
    where Participe.numad=Adherent.numad;
```

Syntaxe d'une requête

```
select <COLS> from <TABS> where <CONDS>;
```

- <COLS> effectue une projection
- <TABS> effectue un produit cartésien
- <CONDS> effectue une sélection. La clause **where** est facultative

Exemple

```
select nom, prenom, intitule
from Adherent, Participe
where Adherent.numad=Participe.numad
  and Adresse='Orleans';
```

Conditions

$$
\begin{aligned}
\langle \text{cond} \rangle &::= \langle \text{bool} \rangle \\
&\mid \texttt{(} \langle \text{cond} \rangle \texttt{)} \\
&\mid \langle \text{cond} \rangle \quad \langle \text{op} \rangle \quad \langle \text{cond} \rangle \\
&\mid \texttt{NOT} \quad \langle \text{cond} \rangle \\
\\
\langle \text{op} \rangle &::= \texttt{AND} \mid \texttt{OR} \\
\\
\langle \text{bool} \rangle &::= \langle \text{comparaison} \rangle
\end{aligned}
$$

il existe de nombreuses autres expressions Booléennes possibles.

Comparaisons

⟨comparaison⟩ ::= ⟨expr⟩ ⟨op⟩ ⟨expr⟩

⟨op⟩ ::=	=	égal
|	<> | !=	différent
|	< | <= | > | >=	ordre
|	**IS NULL**	absence de

⟨expr⟩ ::= ⟨constante⟩
| ⟨colonne⟩ | ⟨table⟩.⟨colonne⟩
| ⟨expr⟩ ⟨opération⟩ ⟨expr⟩

Comparaisons de chaînes

Opérateur de comparaison approximative **LIKE** :

⟨chaine⟩ **LIKE** '⟨motif⟩'

où ⟨motif⟩ est une suite de caractères contenant éventuellement les symboles **%** et _

- **%** remplace n'importe quelle suite de caractères
- _ remplace exactement un caractère

'abc**%**' désigne tous les mots commençant par abc

Expressions

- ⟨opérations⟩ sont standard pour l'arithmétique
- pour les chaînes || désigne la concaténation
- ⟨date⟩ + ⟨entier⟩ ajoute un nombre de jours à une date
- ⟨date⟩ – ⟨entier⟩ enlève un nombre de jours à une date
- ⟨date⟩ – ⟨date⟩ donne le nombre de jours entre deux dates

Renommage

Liste des adhérents qui ont au moins un homonyme :

```
select r1.nom, r1.prenom p1, r2.prenom p2
  from Adherent r1, Adherent r2
 where r1.nom=r2.nom
   and r1.numad<>r2.numad;
```

Renommage : affichage

NOM	P1	P2
Duval	Paul	Jean
Duval	Jean	Paul

Exemple complet

Liste des adhérents habitant Orléans avec leurs activites :

```
select nom, prenom, numact act
  from Adherent a, Participe p
 where a.numad=p.numad
   and adresse='Orleans';
```

Calculs dans le **select**

Les colonnes du **select** peuvent être le résultat d'un calcul :

```
select Upper(nom), (Sysdate - datenais)/365
  from Adherent;
```

Affiche le nom des adhérents en majuscules ainsi que leur age.

Les calculs peuvent se faire entre plusieurs colonnes d'une même ligne :

```
select intitule, prixht*qte
  from commande;
```

Formatage du résultat

- élimination des doublons
- tri des données

Elimination des doublons

Apres une projection, il y a risque de doublons dans le résultat :

```
select ville from Adherent;
```

Ville
Orleans
Tours
Orleans

Elimination des doublons

Elimination des doublons par **DISTINCT** :

```
select DISTINCT ville from Adherent;
```

	Ville
	Orleans
	Tours

Tri des données

Par la clause **`order by`**

$$
\begin{array}{rcl}
\langle \text{order by} \rangle & ::= & \textbf{order by} \quad \langle \text{bycols} \rangle \\
\langle \text{bycols} \rangle & ::= & \langle \text{bycol} \rangle \\
& | & \langle \text{bycol} \rangle, \quad \langle \text{bycols} \rangle \\
\langle \text{bycol} \rangle & ::= & \langle \text{col} \rangle \\
& | & \langle \text{col} \rangle \quad \textbf{desc}
\end{array}
$$

l'ordre par défaut est ascendant (sauf avec **`desc`**). La clause **`order by`** est toujours placée en dernier dans la requête

Exemple

```
select nom, prenom
  from Adherent
 order by nom, prenom;
```

cette requête affiche la liste des adhérents classée dans l'ordre alphabétique par nom puis prenom

Exemple

```
select Activite.intitule
  from Activite, Participe
 where Activite.numact=Participe.numact
   and anneeparticipe=2000
 order by Activite.intitule;
```

affiche la liste classée par ordre alphabétique des activités ayant eu des participants en 2000

Exemple

```
select nom, prenom, datenais
  from Adherent
 order by datenais desc;
```

affiche la liste des adhérents du plus jeune au plus agé