Ressource R207
Sources de données

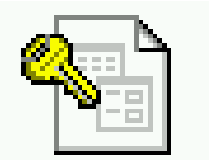
ORACLE
FRANCE

DB2 Information Management Software

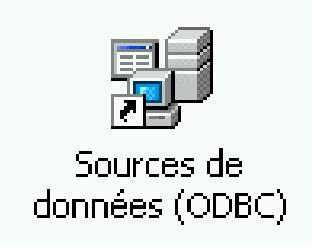
SYBASE

{JSON}

Réseaux & Télécoms



Microsoft SQL Server 2008



IUT de Béziers, dépt. R&T © 2022
http://www.borelly.net/
Christophe.BORELLY@umontpellier.fr

Contenus de la ressource

- Stockage et accès aux données :
  - Système de gestion de données (relationnel/non relationnel) ;
  - Structuration des données : fichiers (CSV, JSON), exemples de sources ouvertes (open data), web scraping ;
  - Sensibilisation à la réglementation française et internationale (CNIL, RGPD).
- Base de données relationnelles :
  - Schéma relationnel d'une base de données ;
  - Sensibilisation aux contraintes d'intégrité ;
  - Création de tables simples ;
- Interrogation de données, ajout et modification de données.
- Lecture d'une documentation technique (UML, diagramme de classes).

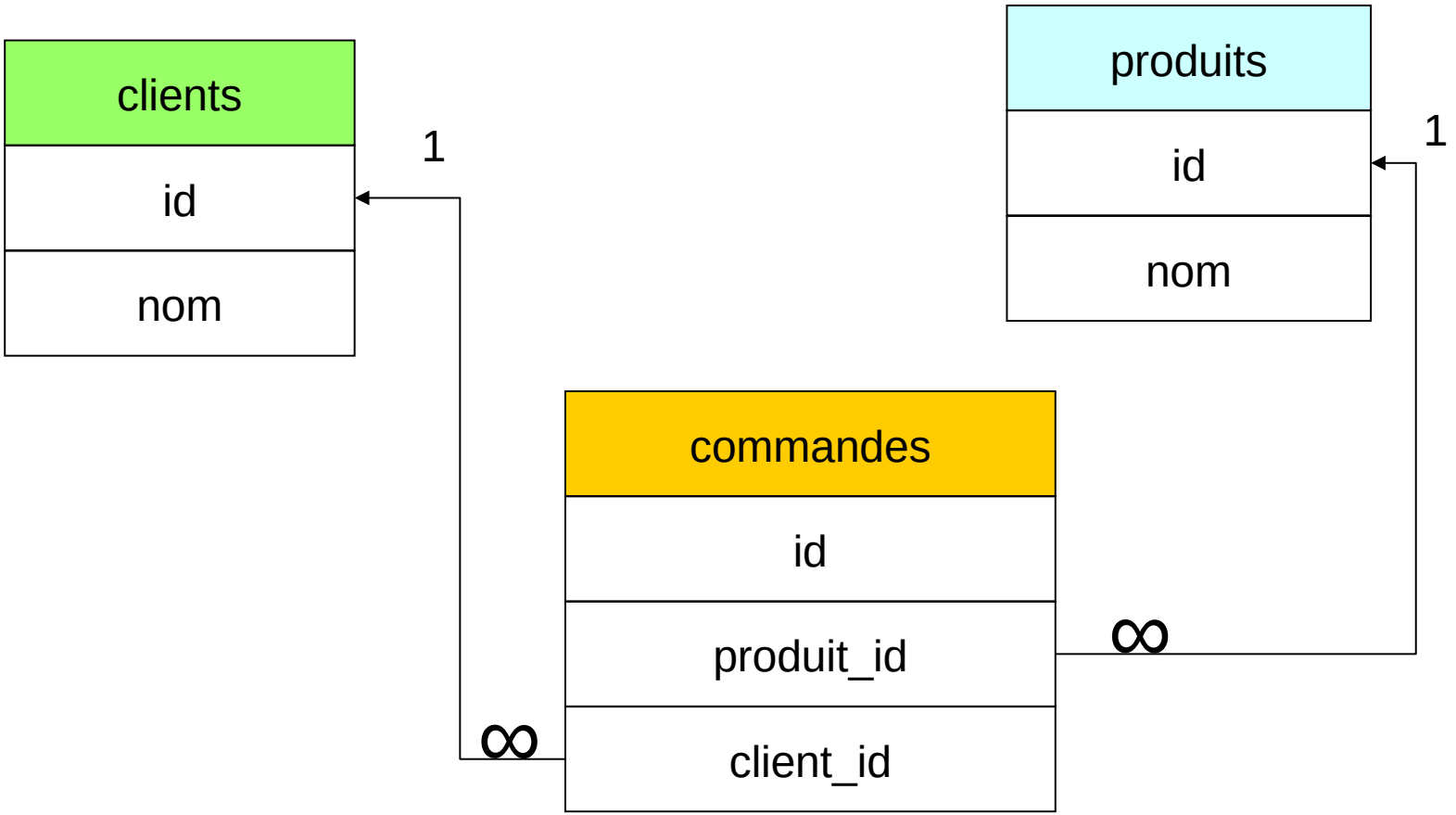
Généralités

Le but d’une base de données :

- Organiser des données (par l’intermédiaire de tables).
- Faciliter la recherche, la mise à jour, la sauvegarde des informations.

Cours 1+1 (1h15), **TD** 3 (1h15), **TP** 3 (2h45)

Schéma conceptuel



Les tables

- Regroupement d'informations sur un thème précis.
- Les champs (colonnes, field)
- Parmi les champs : la clé primaire (un champ à valeur unique)
- Les enregistrements (lignes, row)
- Les indexes permettent d'accélérer les recherches.

users

id	nom	pass
1	titi	kds2d
2	tata	cvBnr
3	toto	kDsfd

Les requêtes (query)

- Une requête est une opération (sélection, insertion, mise à jour, ...) sur une ou plusieurs tables.
- Les vues sont des « raccourcis » d'une requête donnée.
- Les procédures stockées correspondent à un ensemble de requêtes.

Langage SQL (Structured Query Language)

- Normes ANSI SQL 92, ANSI SQL 99.
- Universel pour toute base de données.
- Certaines fonctionnalités ne sont pas supportées suivant les SGBD.
- Certaines fonctionnalités des SGBD sont « propriétaires ».
- Ce cours s'applique plus particulièrement aux bases MySQL/MariaDB.

Les types de données MySQL

- Les chaînes (encadrées par ' ou ")
- Les dates (encadrées par ' ou ")
- Les nombres (sans rien ou encadrés par ' ou ")
- Tous les types peuvent prendre la valeur NULL (pas de valeur)
  - TINYINT, SMALLINT, MEDIUMINT, **INT**, BIGINT
  - FLOAT, **DOUBLE**, DECIMAL
  - DATE, TIME, **DATETIME**, YEAR, TIMESTAMP
  - TINYTEXT, SMALLTEXT, MEDIUMTEXT, TEXT, BIGTEXT
  - CHAR, **VARCHAR**, BLOB, ENUM, SET

Les types dates et heures

- Les types dates et heures sont :
  - DATE 'YYYY-MM-DD' (année-mois-jour)
  - TIME 'hh:mm:ss' (heure:minute:seconde)
  - DATETIME 'YYYY-MM-DD hh:mm:ss'
  - TIMESTAMP 'YYYYMMDDhhmmss'
  - YEAR 'YYYY'

Les types CHAR et VARCHAR

- Les types CHAR et VARCHAR sont similaires, mais diffèrent dans la manière dont ils sont stockés et récupérés (les espaces finaux sont retirés des valeurs issues des champs de type CHAR lors de la lecture).
- La taille maximum est de 255 caractères.

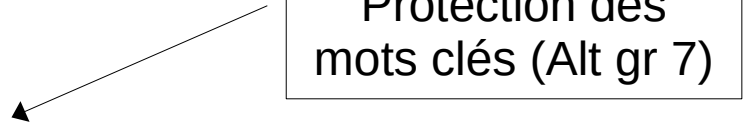
```
Valeur      CHAR(4) Taille   VARCHAR(4)  Taille
''          '    '  4 octets ''          1 octet
'ab'        'ab  '  4 octets 'ab'        3 octets
'abcd'      'abcd'  4 octets 'abcd'      5 octets
'abcdefgh' 'abcd'  4 octets 'abcd'      5 octets
```

Jeux de caractères et collation MySQL

- Un jeu de caractères est un ensemble de symboles et de codes.
- Une collation est un ensemble de règles permettant la comparaison de caractères dans un jeu.
- SHOW CHARACTER SET;
- SHOW COLLATION LIKE 'utf8%';

Création d'une base MySQL

- **CREATE DATABASE** bibli;
- CREATE DATABASE IF NOT EXISTS annuaire;
- CREATE DATABASE `users` DEFAULT CHARACTER SET latin1;
- CREATE DATABASE `users` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;



Autres commandes sur les bases MySQL

- `SHOW DATABASES;`
- `SHOW CREATE DATABASE \`test\`;`
- `ALTER DATABASE annuaire DEFAULT CHARACTER SET utf8;`
- `ALTER DATABASE \`users\` DEFAULT CHARACTER SET latin1 COLLATE latin1_bin;`
- `DROP DATABASE \`users\`;`
- `DROP DATABASE IF EXISTS annuaire;`

Création d'une table MySQL

- ```
  CREATE TABLE test.t1
  (a INT(11) UNSIGNED AUTO_INCREMENT NOT NULL,
   b VARCHAR(4) DEFAULT 'toto',
   PRIMARY KEY(a));
  ```
- ```
  USE test;
  ```
- ```
  CREATE TABLE IF NOT EXISTS t2
  (s ENUM('oui','non') DEFAULT NULL,
   t DATETIME NOT NULL,
   PRIMARY KEY (t));
  ```
  ```

Affichage de la structure d'une table

- SHOW TABLES FROM test;

```
+----------------+
| Tables_in_test |
+----------------+
| t1             |
| t2             |
+----------------+
```

- SHOW CREATE TABLE test.t2;

```
+-------+-----------------------------------------------+
| Table | Create Table                                  |
+-------+-----------------------------------------------+
| t2    | CREATE TABLE `t2` (                           |
|       | `s` enum('oui','non') default NULL,           |
|       | `t` datetime NOT NULL,                        |
|       |  PRIMARY KEY  (`t`)                           |
|       | ) ENGINE=MyISAM DEFAULT CHARSET=utf8          |
+-------+-----------------------------------------------+
```

- SHOW COLUMNS FROM `test`.t2;

```
+-------+-------------------+------+-----+---------+-------+
| Field | Type              | Null | Key | Default | Extra |
+-------+-------------------+------+-----+---------+-------+
| s     | enum('oui','non') | YES  |     | NULL    |       |
| t     | datetime          | NO   | PRI |         |       |
+-------+-------------------+------+-----+---------+-------+
```

Modification des tables MySQL

- `ALTER TABLE test.t1 RENAME test.t3;`
- `ALTER TABLE test.t3 ADD c INT(11) AFTER b;`
-
```
ALTER TABLE test.t3
   MODIFY a TINYINT NOT NULL,
   CHANGE b bb CHAR(20);
```
- `ALTER TABLE test.t3 DROP COLUMN c;`
- `RENAME TABLE `iut`.`users` TO `iut`.`users2`;`
- `RENAME TABLE `iut`.`users` TO `iut`.`users2`, `bibli`.`livres` TO `bibli`.`bouquins`;`

Ajout de données

- **INSERT INTO** t1 (a,b)
  VALUES (15,'toto');
- INSERT INTO `test`.`t2` (w,x,y,z)
  VALUES (NOW(),16,x*2,NULL);
- INSERT INTO db.t3 (a,b)
  VALUES ("tata",2),('l\'ile','5');
- INSERT INTO db.t3 SET a='toto',b=2;

Mise à jour des données

- **UPDATE** personnes SET age=12, prenom='titi' WHERE nom='toto';
- UPDATE personnes SET age=age+1;
- UPDATE log SET day=NOW() WHERE id=4;
- UPDATE produits,devis SET produits.prix=devis.prix WHERE produits.id=devis.id;

Sélection de données (1)

- `SELECT * FROM `test`.`t2`;`

```
+------+---------------------+
| s    | t                   |
+------+---------------------+
| oui  | 2007-05-29 15:00:25 |
| non  | 2007-05-29 00:00:00 |
| NULL | 2007-05-29 15:01:11 |
+------+---------------------+
```

- `SELECT s FROM `test`.`t2`;`

```
+------+
| s    |
+------+
| oui  |
| non  |
| NULL |
+------+
```

Sélection de données (2)

- `SELECT t,s FROM `test`.`t2`;`

```
+---------------------+------+
| t                   | s    |
+---------------------+------+
| 2007-05-29 15:00:25 | oui  |
| 2007-05-29 00:00:00 | non  |
| 2007-05-29 15:01:11 | NULL |
+---------------------+------+
```

- `SELECT t AS Heure,s AS Val FROM `test`.`t2`;`

```
+---------------------+------+
| Heure               | Val  |
+---------------------+------+
| 2007-05-29 15:00:25 | oui  |
| 2007-05-29 00:00:00 | non  |
| 2007-05-29 15:01:11 | NULL |
+---------------------+------+
```

Conditions

- `SELECT x FROM t1 WHERE x>10;`
- `SELECT y FROM t1 WHERE x IS NOT NULL;`
- `SELECT x FROM t1 WHERE x>1 AND y='to';`
- ```
  SELECT n FROM t1
  WHERE n LIKE 'a%' OR n LIKE '_a%';
  ```
- ```
  SELECT t1.*,t2.salaire
  FROM employes AS t1,infos AS t2
  WHERE t1.name=t2.name;
  ```

Nombre de résultats

- Retourne les 5 premiers enregistrements :

```
SELECT * FROM t1 LIMIT 5;
```

- Retourne les enregistrements 6 à 15 :

```
SELECT * FROM t1 LIMIT 5,10;
```

Tri des données

- ```
  SELECT college,region,ville
  FROM lycee
  ORDER BY region;
  ```
- ```
  SELECT college,region,ville
  FROM lycee
  ORDER BY region ASC,ville DESC;
  ```

Regroupement de valeurs identiques

- ```
  SELECT ville AS N FROM personnes
  GROUP BY ville;
  ```
- ```
  SELECT pays,ville,SUM(nb) AS t
  FROM recencement
  GROUP BY pays,ville;
  ```

Comptage

- COUNT(xxx) : Retourne le nombre de valeurs différentes de NULL du champ xxx :

```
SELECT ville,COUNT(adresse)
   FROM stages
   GROUP BY ville ASC;
```

- COUNT(*) est un peu différente, car elle retourne le nombre de lignes, même si elles contiennent NULL.

```
SELECT COUNT(*) FROM bibli.livres;
```

Effacement des données

```
DELETE FROM t1 WHERE t1.id>10;
DELETE FROM t1 WHERE id=12 OR id=15;
```

- Si la commande DELETE inclut la clause ORDER BY, les lignes sont effacées dans l'ordre spécifiée par cette clause. Elle n'est vraiment utilisée que lorsqu'elle est couplée avec la clause LIMIT.

```
DELETE FROM log WHERE user='jdoe'
ORDER BY dateInscription LIMIT 1;
```

Insertions avancées

- La requête INSERT ... SELECT permet d'insérer rapidement dans une table, un grand nombre de lignes d'une ou plusieurs autres tables.

```
INSERT INTO test.t2 (id,nom)
  SELECT test.t1.order_ID,test.t1.name
  FROM test.t1 WHERE t1.order_ID>100;
```

- Vous pouvez trouver la valeur utilisée pour une colonne AUTO_INCREMENT en utilisant la fonction LAST_INSERT_ID()

```
USE test;
INSERT INTO t2 SET b='toto';
INSERT INTO t3 SET ref=LAST_INSERT_ID();
```

Liaisons entre deux tables

users

id	nom
1	titi
2	tata
3	toto

commandes

id	produit	users_id
6	Caméra	2
9	Voiture	1
14	PC	1
25	Téléphone	2

Liaisons entre deux tables

```
SELECT nom,produit
FROM users,commandes
WHERE users.id=commandes.users_id;
```

```
SELECT nom,produit
FROM users INNER JOIN commandes
ON users.id=commandes.users_id;
```

nom	produit
tata	Caméra
titi	Voiture
titi	PC
tata	Téléphone

Liaison à gauche

- Renvoi toutes les lignes de la première table (même si il n'y a pas de correspondance dans la seconde table)

```
SELECT nom,produit
FROM users
LEFT JOIN commandes
ON
users.id=commandes.users_id
```

nom	produit
titi	Voiture
titi	PC
tata	Caméra
tata	Téléphone
toto	*NULL*

Liaison à droite

- Renvoi toutes les lignes de la seconde table (même si il n'y a pas de correspondance dans la première table)

```
SELECT nom,produit
FROM users
RIGHT JOIN commandes
ON
users.id=commandes.users_id
```

nom	produit
tata	Caméra
titi	Voiture
titi	PC
tata	Téléphone

Effacement avancés

- Exemple d'effacement des données de 2 tables (t1 et t2) à partir de conditions sur une troisième (t3):

```
DELETE t1,t2 FROM t1,t2,t3 WHERE
t1.id=t2.id AND t2.id=t3.id AND t3.id>50;
```

- Autre syntaxe équivalente :

```
DELETE FROM t1,t2 USING t1,t2,t3 WHERE
t1.id=t2.id AND t2.id=t3.id AND t3.id>50;
```

Union de résultats

- Depuis MySQL 4.0.0, UNION est utilisé pour combiner le résultat de plusieurs requêtes SELECT en un seul résultat.

```
SELECT nom FROM personnes
UNION
SELECT name FROM users WHERE age>25;
```

Sous requêtes

- Trouver toutes les valeurs de la table t1 qui sont égales au maximum de la valeur dans la table t2 :

```
SELECT x FROM t1 WHERE x=(SELECT
MAX(y) FROM t2);
```

- Trouver toutes les lignes de la table t1 qui contiennent une valeur qui apparaît deux fois :

```
SELECT * FROM t1 WHERE 2=(SELECT
COUNT(x) FROM t1);
```

Téléchargements

- http://www.mysql.com/ ou alors http://mariadb.org/
- http://httpd.apache.org/
- http://www.php.net/
- http://sourceforge.net/projects/phpmyadmin/
- http://www.wampserver.com/
  - **WAMP** : Apache + MySQL + PHP + PhpMyAdmin
- http://sourceforge.net/projects/xampp/
  - **XAMPP** : Apache + MySQL + PHP + Perl + PhpMyAdmin

Références

- Norme ANSI-ISO-9075-x-1999
- http://dev.mysql.com/doc/refman/5.0/fr/
- http://www.w3schools.com/sql/