Apprendre le langage SQL par l'exemple

Partie 2 : le DML

Présentation

Le DML, c'est essentiellement quatre instructions :

- **INSERT**
  Ajout d'enregistrement à une table
- **UPDATE**
  Mise à jour d'enregistrement(s)
- **DELETE**
  Suppression d'enregistrement(s)
- **SELECT**
  Extraction de données issues d'une ou plusieurs tables ou vues

Transactions

Le mode transactionnel est disponible en InnoDB.

Les modifications apportées à une table par l'emploi des commandes INSERT, UPDATE ou DELETE ne sont visibles des autres utilisateurs qu'à partir de l'exécution de l'instruction **COMMIT** par l'utilisateur qui a procédé aux modifications.

À l'occasion de ces modifications, MySQL pose des **verrous** sur ligne ou sur table. Ils empêchent les autres utilisateurs d'accéder aux données modifiées, dans un souci de **cohérence** des données du système d'information.

Les verrous posés au cours de la transaction sont détruits par l'exécution du COMMIT ou du **ROLLBACK**. ROLLBACK remet la base dans l'état qui prévalait avant les modifications.

Instructions liées aux transactions

Mode transactionnel

L'une ou l'autre de ces deux commandes ouvre une transaction. Elle est fermée par un COMMIT ou un ROLLBACK.

```
START TRANSACTION
BEGIN [WORK]
```

Mode transactionnel

```
COMMIT [WORK] :   valide la transaction
ROLLBACK [WORK] :   invalide la transaction
```

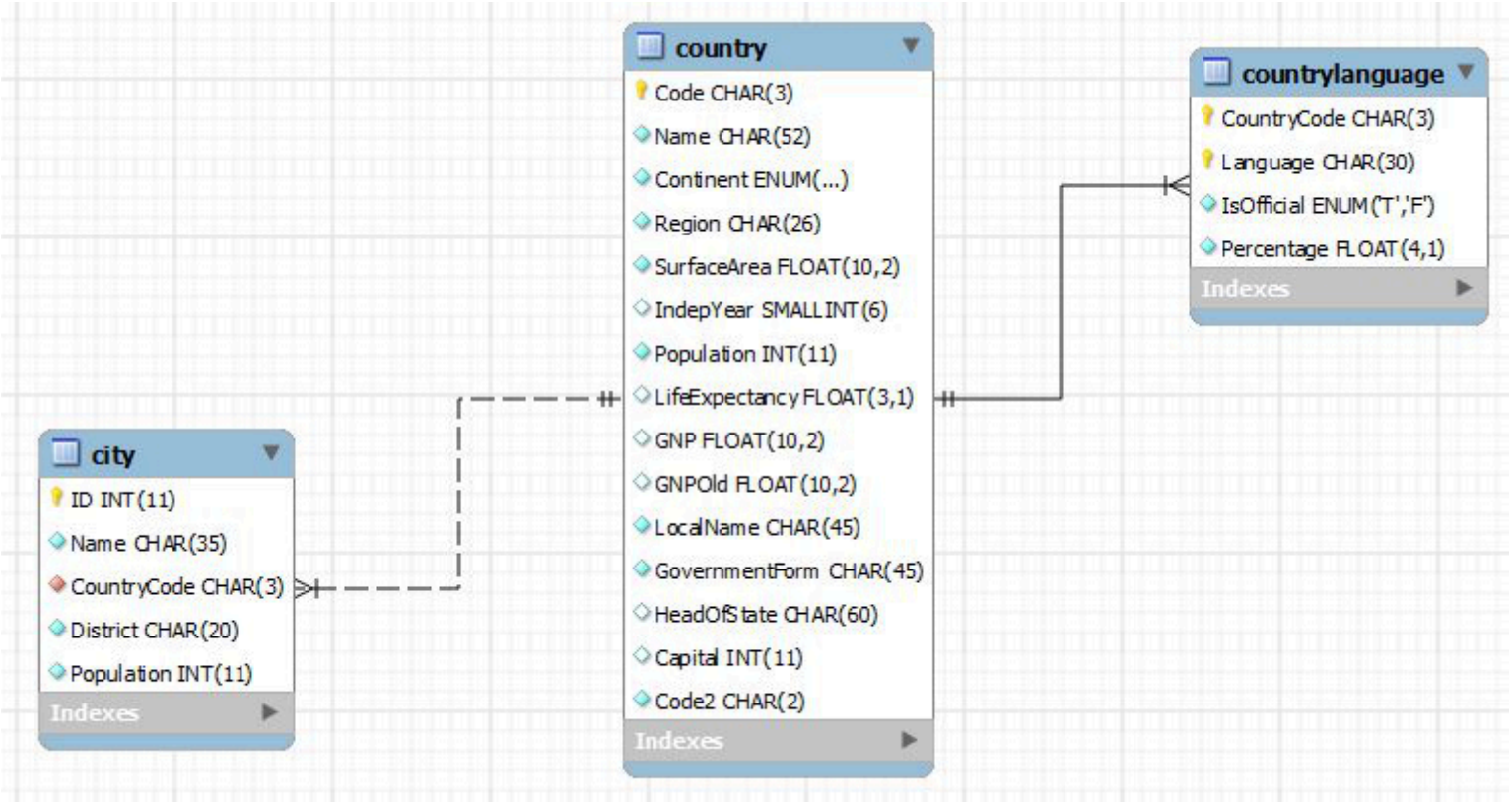
Mode auto-commit

Par défaut, MySQL est en auto-commit. Les transactions sont automatiquement validées. Vous pouvez changer le paramètre avec la commande :

```
SET auto-commit = {0 | 1}
```

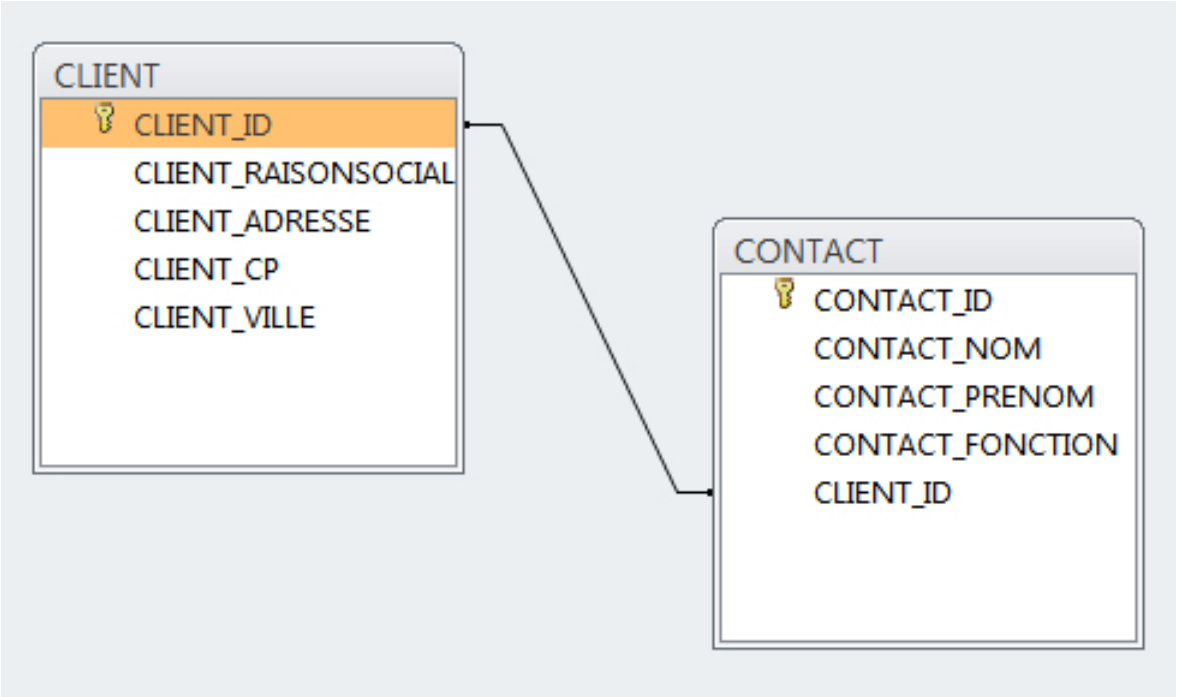
Les tables du schéma World

MySQL dispose de nombreuses bases d'exemple accessibles à partir de ce lien !



Les tables du schéma CRM

C'est l'exemple sur lequel nous nous sommes appuyés dans le support consacré à l'étude du DDL.



INSERT

```
INSERT INTO world.country (country_id,
country_iso_code, name, country_subregion,
country_subregion_id, region, region_id,
country_total, country_total_id)

VALUES ('52999', 'MA', 'Maroc', 'Africa', '52792',
'Africa', '52800', 'World total', '52806')
```

Ou bien, lorsque toutes les colonnes sont renseignées dans l'ordre où elles sont présentées dans la table…

```
INSERT INTO world.country
VALUES ('52999', 'MA', 'Maroc', 'Africa', '52792',
'Africa', '52800', 'World total', '52806')
```

INSERT en présence d'un compteur

Lorsque la table possède un compteur (séquence + trigger), évitez de le renseigner !

```
INSERT INTO crm.client (client_raisonsociale,
client_adresse, client_cp, client_ville)
VALUES ('Dsfc', '5 route de Saint Paul', 27800,
'Saint-Eloi-de-Fourques')
```

Multi-affectations avec INSERT

L'ordre SQL INSERT sous MySQL vous permet d'ajouter plusieurs lignes en une seule instruction :

```
INSERT INTO crm.client(client_raisonsociale,
client_adresse, client_cp, client_ville)
VALUES('Dsfc', '5 route de Saint Paul', 27800,
'Saint-Eloi-de-Fourques'),
('Denis Szalkowski', '5 route de Saint Paul', 27800,
'Saint-Eloi-de-Fourques')
```

DELETE

L'instruction DELETE s'emploie à titre principal avec la clause WHERE. Pour supprimer toutes les données d'une table de manière irréversible, vous disposez de l'instruction TRUNCATE. Pour valider l'effacement d'enregistrement, vous devez procéder, ensuite, à un COMMIT. **Plus dangereux, le TRUNCATE est beaucoup plus rapide !!!**

Mode transactionnel

```
DELETE FROM crm.contact;
COMMIT;
```

Mode non transactionnel

```
TRUNCATE TABLE crm.contact;
```

Utilisation de la clause WHERE

Suppression à partir de la valeur de la clé

```
DELETE FROM crm.client
WHERE client_id = 1;
COMMIT;
```

Suppression de plusieurs lignes

```
DELETE FROM crm.client
WHERE client_cp BETWEEN 93000 AND 93999;
COMMIT;
```

UPDATE

Mise à jour des valeurs des colonnes d'une table

```
UPDATE crm.client
SET
client_raisonsociale=UPPER(client_raisonsociale),
client_ville=UPPER(client_ville);
COMMIT;
```

Mise à jour de lignes

```
UPDATE crm.client
SET client_raisonsociale='Denis Szalkowski'
WHERE
client_raisonsociale = 'Dsfc' AND client_cp=27800;
COMMIT;
```

SELECT : la syntaxe de base

L'instruction SELECT permet d'afficher le contenu d'une table. Tapez votre instruction sur plusieurs lignes, en indentant le code SQL, afin de le rendre plus lisible. L'instruction ci-dessous affiche toutes les colonnes et toutes les lignes d'une table.

```
SELECT
  *
FROM
 world.country
```

Oracle n'est pas sensible à la casse au niveau des instructions et des clauses utilisées : SELECT et select, d'une part, FROM et from, d'autre part, seront interprétés de la même façon. Sous Oracle, nous avons pour habitude de taper instructions et clauses en majuscules. Les expressions introduites par l'utilisateur sont en général en minuscules.

Commentaires

Pour rendre plus compréhensible les instructions SQL, vous pouvez employer les commentaires dans vos fichiers.

```
/*
Commentaire en bloc,
sur plusieurs lignes
*/
```

SELECT : affichage partiel

Vous pouvez, en les spécifiant, visualiser, du contenu de la table, le contenu de certaines colonnes.

```
SELECT
 region,
 name
FROM
 world.country
```

Alias de colonnes

Les applications utilisant des bases de données présentent très souvent des identifiants de colonnes en anglais. Vous pouvez associer un alias à ces colonnes.

```
SELECT
 region AS continent,
 name AS pays
FROM
 world.country
```

La clause AS est totalement facultative.

Utilisation des guillemets dans les alias de colonnes

Au niveau des alias de colonnes, MySQL n'est pas sensible à la casse. Pour obliger MySQL à afficher minuscules, majuscules ou à afficher des caractères spéciaux et des espaces dans les alias de colonnes, vous devez les encadrer de guillemets !

```
SELECT
 region "Continent",
 name "Pays"
FROM
 world.country
```

Tri des données

Le tri des lignes d'une table peut se faire de manière croissante ou décroissante, sur la valeur d'une ou de plusieurs colonnes. L'existence d'un index sur la colonne du tri accélère l'exécution de l'instruction SQL.

```
SELECT
 region,
 name
FROM
 world.country
ORDER BY
 region DESC,
 name ASC
```

Spécificité de l'emploi de la clause ORDER BY

La clause ORDER BY est la seule clause associée à L'ordre SQL qui peut utiliser les alias de colonnes en lieu et place des noms de colonnes.

```
SELECT
 region continent,
 name pays
FROM
 world.country
ORDER BY
 continent DESC,
 pays ASC
```

La clause DISTINCT

Pour éliminer l'affichage des doublons portant sur une, plusieurs ou la totalité des colonnes, vous pouvez utiliser la clause DISTINCT. Son utilisation trie les données dans l'ordre de présentation des colonnes définies dans l'instruction SELECT.

```
SELECT DISTINCT
 region
FROM
 world.country
```

La clause LIMIT

C'est un des intérêts de MySQL. La clause LIMIT permet de limiter le nombre de lignes affichées. Le 1er paramètre indique le numéro de ligne à partir de laquelle vous demandez à afficher les données. Le 2e paramètre représente le nombre de lignes à afficher.

```
SELECT
 region
FROM
 world.country
LIMIT 10,3
```

Fonctions courantes

L'ordre SELECT permet l'utilisation de fonctions au niveau des colonnes et aussi au niveau de la clause WHERE.

- UPPER, LOWER : convertit en majuscules, minuscules
- REGEXP : permet la manipulation d'expressions régulières
- SYSDATE, NOW : SELECT SYSDATE [FROM DUAL] affiche la date du jour
- CASE, IF : gère l'affichage de contenus sur conditions
- CAST : opère la conversion de type (numérique en alpha-numérique)
- CONCAT : permet la concaténation des champs
- ROUND : permet d'arrondir

Ces fonctions sont utilisables également au niveau de la clause WHERE.
Les champs présentés dans l'ordre SELECT peuvent être des champs calculés.

La clause WHERE

La clause WHERE permet de filtrer les lignes selon des conditions portant sur les valeurs de colonnes. Vous pouvez utiliser les opérateurs IS, OR, NOT, OR, AND, BETWEEN, LIKE ou IN.

```
SELECT
  ROUND(gnp/population*1000000/1.37/12) as pnb,
  name AS pays,
  population
FROM
  world.country
WHERE
  population>10000000
```

Expression conditionnelle

L'expression conditionnelle CASE... WHEN...ELSE... END possède deux syntaxes.

```
SELECT
  name AS pays,
  CASE REGION
    WHEN 'Southern Europe' THEN 'Europe du Sud'
    WHEN 'Eastern Europe' THEN 'Europe de l\'Est'
    WHEN 'Western Europe' THEN 'Europe de l\'Ouest'
    ELSE 'Ailleurs'
  END AS region1,
  CASE
    WHEN REGION='Southern Europe' THEN 'Europe du Sud'
    WHEN REGION='Eastern Europe' THEN 'Europe de l\'Est'
    WHEN REGION='Western Europe' THEN 'Europe de l\'Ouest'
    ELSE 'Ailleurs'
  END AS region2
 FROM world.country
WHERE
  region LIKE '%Europe%'
ORDER BY
  pays;
```

Utilisation des expressions régulières

Les expressions régulières nous offrent d'immenses possibilités quant à l'extraction d'informations à partir de masques portant sur des chaînes de caractères. Sous MySQL, vous disposez des fonctions REGEXP ou RLIKE.

```
SELECT
  *
FROM
  world.city
WHERE
  district REGEXP '(west|east)('
```

Les regroupements statistiques

Le SQL comprend plusieurs fonctions de regroupement statistiques :

- COUNT,
- MIN, MAX,
- SUM,
- AVG,
- STDDEV,
- VARIANCE.

Compter le nombre d'enregistrements d'une table

Les fonctions de regroupement peuvent s'employer simplement.

```
SELECT
 COUNT(*)
FROM
 world.country
```

Vous devez compter sur une colonne renseignée, comme la clé, par exemple.

```
SELECT
 COUNT(code)
FROM
 world.country
```

Utilisation du GROUP BY

Vous pouvez utiliser la clause GROUP BY à la manière d'un DISTINCT.

```
SELECT
  CountryCode
FROM
  world.city
GROUP BY
  CountryCode
```

Pour compter le nombre de villes par pays :

```
SELECT
  CountryCode pays,
  COUNT(Code) total
FROM
  world.city
GROUP BY
  CountryCode
```

Dans la clause GROUP BY, vous devez indiquer tous les champs figurant dans le SELECT, à l'exception de ceux sur lesquels portent les statistiques.

La clause WHERE dans les regroupements

Elle permet d'appliquer aux statistiques un filtre sur les enregistrements sur lesquels porteront les indicateurs statistiques.

```
SELECT
 region,
 governmentform,
 ROUND(SUM(GNP)/SUM(population)*100000,2) pnbparhab
FROM
 world.country
WHERE
 region LIKE '%europe'
GROUP BY
 region,
 governmentform
```

La clause HAVING

Elle permet d'opérer un filtre sur l'indicateur de regroupement utilisé au niveau de l'ordre SELECT.

```
SELECT
  region,
  governmentform,
  ROUND(SUM(GNP)/SUM(population)*100000,2) pnbparhab
FROM
  world.country
WHERE
  region LIKE '%europe'
GROUP BY
  region,
  governmentform
HAVING
  SUM(GNP)/SUM(population)*100000>2000
```

Les requêtes imbriquées

L'utilisation des requêtes imbriquées suppose l'existence d'un champ commun aux deux requêtes. Dans l'exemple ci-dessous, il s'agit de la clé primaire de la table "parent", *world.country*, et la clé étrangère de la table "enfant", *world.city*.

```
SELECT
  name ville,
  population
FROM
  world.city
WHERE
  CountryCode IN
  (

    SELECT DISTINCT          Code        FROM                world.country
  WHERE          continent LIKE '%europe%' );
```

Les opérateurs ensemblistes

Contrairement à Oracle ou SQL Server, nous ne disposons sous MySQL d'un seul opérateur ensembliste : UNION. Il permet de réaliser très simplement les requêtes, sans compliquer outrageusement le contenu de la clause WHERE. Ils peuvent être coûteux !

Ils permettent aussi d'agréger des contenus issus de colonnes différentes, à la condition que le nombre et le type de colonnes soient identiques dans les différentes requêtes.

Les jointures

Naguère réalisées au travers de la clause WHERE, les jointures, depuis la normalisation de 1999, se réalisent au niveau de la clause FROM utilisée au niveau de l'instruction SELECT !

Nous sommes amenés à utiliser les jointures lorsque nous voulons afficher des données issues de différentes tables. La réalisation d'une jointure entre deux tables exige la présence d'un champ du même type dans les deux tables ! En général, la jointure se fait sur la clé primaire d'un côté et sur la clé étrangère de l'autre, selon les caractéristiques définies au travers du modèle relationnel et du modèle physique des données.

Les équijointures

Dans cet exemple, nous affichons la liste des régions de WORLD.COUNTRY qui pratiquent les langues françaises, allemandes et anglaises, définies dans la table WORLD.COUNTRYLANGUAGE.

```
SELECT
 co.region,
 cl.language

FROM
 world.country AS co
 INNER JOIN world.countrylanguage as cl
 ON co.code = cl.CountryCode
WHERE
 cl.language IN ('french', 'german', 'english');
```

Jointure externe

Il peut y avoir des enregistrements de WORLD.COUNTRY qui n'aient pas de correspondance dans WORLD.CITY. La jointure externe permet d'obtenir tous les enregistrements de l'une des deux tables sans qu'ils aient d'enregistrements corrélés dans l'autre. Le mot OUTER est parfaitement facultatif.

```
SELECT
  co.name AS pays,
  CASE
    WHEN SUM(ci.population) is NULL then 0
    ELSE SUM(ci.population)
  END AS population
FROM
  world.country AS co
  LEFT JOIN world.city AS ci
  ON co.code = ci.CountryCode
GROUP BY
  co.name;
```

Tester les enregistrements sans correspondance

Pour obtenir les enregistrements de la table WORLD.COUNTRY sans correspondance dans WORLD.CITY, il suffit d'utiliser la clause WHERE dans l'instruction.

```
SELECT
  co.name AS pays,
  SUM(ci.population) AS population
FROM
  world.country AS co
  LEFT JOIN world.city AS ci
  ON co.code = ci.CountryCode
WHERE
  ci.population IS NULL
GROUP BY
  co.name;
```

LEFT ou RIGHT ?

Vous êtes gaucher ou droitier ? Adaptez alors votre syntaxe à votre sens de lecture ! ;+) L'instruction suivante est identique, en tout point, à la jointure précédente réalisée avec un LEFT.

```
SELECT
 co.name AS pays,
 CASE
  WHEN SUM(ci.population) is NULL then 0
  ELSE SUM(ci.population)
 END AS population
FROM
 world.city AS ci
 RIGHT JOIN world.country AS co
 ON co.code = ci.CountryCode
GROUP BY
 co.name;
```

FULL ?

Il est totalement impossible sous MySQL de combiner RIGHT et LEFT dans la même requête. Par ailleurs, l'opérateur FULL OUTER JOIN n'existe toujours pas sur MySQL. Il faut donc utiliser l'opérateur UNION pou pouvoir combiner deux requêtes LEFT et RIGHT, qui, au final, produiront le même résultat que le FULL.

Effacer le spam dans WordPress

Une requête très utile qui vous débarrassera du pourriel dans WordPress !

```
DELETE FROM wp_comments
WHERE
(
    comment_content
    REGEXP
    '([[:space:]]|[[:punct:]]|>|^)(buy|cheap|cheapest|estore|louis vuitton|
order|viagra)([[:space:]]|[[:punct:]]|$|<)'
OR
    comment_author_url
    REGEXP
    '(ciprofloxacin|kamagra|viagra)'
)
AND
    comment_approved<>'1'
;
```

Débloquer votre compte dans WordPress

La première requête permet de réinitialiser votre mot de passe. La seconde débloque votre compte dans le cas où vous utilisez le plugin User Locker.

```
UPDATE wp_users
SET
  user_pass = MD5( 'mon_nouveau_mot_de_passe' ),
  user_activation_key=''
WHERE user_login = 'mon_login';

UPDATE wp_usermeta
SET
  meta_value=0
WHERE meta_key='wp_ul_locked';
```