SQL : Structured Query Language

Développé initialement par IBM, créé en 1974 et normalisé en 1987

Le SQL est un langage normalisé servant à exploiter des bases de données relationnelles (stockage de l'information/données décomposées et organisées dans des tables).

On distingue 4 groupes de commandes :

- DDL data definition language
CREATE, ALTER, DROP, RENAME

-DQL data query language
SELECT

- DML data manipulation language
INSERT, DELETE, UPDATE

-DCL data control language
GRANT, REVOQUE, COMMIT, ROLLBACK

MySQL

MySQL est un serveur de base de données. Les bases de données contiennent des tables. Plusieurs utilisateurs peuvent être définis ayant chacun des droits spécifiques sur une ou des bases de données, on peut préciser des privilèges spécifiques pour une table une action SELECT, UPDATE, ...

Nous utiliserons MySQL en créant des bases de données en UTF8 avec la collation utf8_general_ci (ordre des caractères dans un jeu de caractères données ici l'UTF8).

utf8_general_ci est insensible à la casse, donc plus rapide CI = CASSE INSENSIBLE

Les types en SQL

INT sur 4 octets (1 octet=8 bits par exemple 10100011), nombre compris entre -2147483648 et 2147483648
Attention, si la valeur que l'on enregistre dépasse le max ou le min de cet intervalle, alors MySQL met son max ou son min par défaut.

Exercice
créez la table foo dans la base de donnée db_foo, connectez avant au serveur de base de données
>CREATE DATABASE `db_foo` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
>CREATE TABLE `foo` (bar int);
Et essayez d'insérer la valeur suivante 9999999999 ;

BIGINT 8 octets => -9223372036854775808 et 9223372036854775808

TINYINT 1 octet => -127 à 127

TINYINT UNSIGNED 1 octet => 0 à 255

DECIMAL(n,v) pour stocker des nombres sur n chiffres significatifs ayant v chiffres après la virgule:

Exercice
Créez dans la table foo la colonne baz avec le type DECIMAL ayant 5 chiffres en tout dont 3 après la virgules, non nul.
Remarque pour modifier la structure de la table on utilisera la commande suivante :
ALTER TABLE foo ADD baz [TYPE] ;

Qu'arrive-t-il si on essaye d'insérer la valeur suivante pour le colonne que l'on vient de créer :

>INSERT INTO foo (baz) VALUES (70.9979) ;

CHAR(x) et VARCHAR(x)
Pour stocker un texte court x <255 octets on utilise CHAR et VARCHAR

CHAR(x) Stocke (réserve) x octets quelle que soit la valeur que l'on place dans la colonne ayant ce type.
Attention, donc MySQL réserve un nombre d'octets fixe pour chaque champ.

VARCHAR(x) stocke jusqu'à x octets (0 à x octets) selon la taille du mot à stocker. Cette option alloue donc dynamiquement le nombre d'octets requis =< x.

Pour les textes supérieurs à 255 octets on utilise le type

TEXT sur 2^16 octets

BLOB est un type binaire capable de stocker des objets binaires de grande taille. Ici les chaînes de caractères seront sensibles à la casse...Nous ne l'aborderons pas dans le cours.

ENUM Définit un ensemble de caractères autorisés de type chaîne de caractères :
title ENUM('foo', 'bar', 'baz')

DATE
Attention au format
AAAA-MM-JJ ~ AAMMJJ ~ AAAA/MM/JJ ~ AAAA%MM%JJ

Exercice
Créez un champ date_crea dans la table foo et ajouter la ligne suivante :
INSERT INTO foo (date_crea) VALUES ('001202') ;
Quelle date avez-vous rentrée?

DATETIME C'est le type que l'on utilise en général pour les dates de nos posts en base de données.
AAAA-MM-JJ HH:MM:SS

On ne parlera pas du TIMESTAMP de MySQL (mal implémenté), sachez seulement qu'en informatique c'est pour une date donnée « t » le temps en secondes qui s'est écoulé depuis la naissance d'UNIX (1 janvier 1970, 0h0min0s).

DDL (data definition language)
CREATE, ALTER, DROP, RENAME

Convention de nommage pas de caractères spéciaux et d'espace, on écrit les noms comme suit en général, notez que l'on ne met pas de « s » aux noms des bases de données :
db_foo

Petite remarque sur les moteurs de base de données :

ENGINE=InnoDB ou MyISAM

MyISAM est plus rapide et a été amélioré dans ses nouvelles versions, on ne peut mettre que des clefs primaires.

InnoDB fournit un moteur de base de données transactionnelle (compatible ACID), la possibilité de créer des clefs étrangères (foreign key), ... En fait ce moteur est très utilisé dans les gros sites devant optimiser les performances des accès aux données. (InnoDB est sous licence GNU/GPL avec certaines restrictions pour des options du moteur sous licence commerciale voir MySQL Pro).

DEFAULT et NOT NULL

DEFAULT indique une valeur par défaut si on ne précise rien à l'insertion d'une ligne pour cette colonne

Exercice
Ajoutez une colonne status à la table foo de type ENUM les valeurs possibles étant 'publish' et 'unpublish' et la valeur par défaut 'unpublish'.
Vérifiez que la valeur inséré est bien unpublish si on ne précise pas la valeur de status.

Quelques commandes utilises ;

modification de la structure :

ALTER TABLE foo MODIFY baz INT UNSIGNED ;

suppression d'une colonne

ALTER TABLE foo DROP baz ;

NOT NULL si aucune valeur n'est renseignée pour le champ, une valeur par défaut est ajoutée : '' ou 0, ... selon le type de la colonne.
NULL si aucune valeur n'est ajoutée dans la colonne, MySQL met le type NULL dans le champ. Cela est utile dans certains cas, par exemple note de classe, sexe des animaux à la naissance,...

DQL : (data query language)
Interrogation des données SELECT

On passe beaucoup de temps en SQL à faire des SELECT, MySQL est d'ailleurs bien programmé pour faire ce type de commande, lorsqu'on fait un select on fait de l'extraction de données.

Projection : on ne sélectionne ci-dessous que la colonne ref de la table Foo :
SELECT ref FROM foo ;

Restriction : on utilise un where pour n'extraire qu'un sous ensemble de données de la table :
SELECT * FROM WHERE foo_id=1 ;

Si on fait un SELECT sur plus d'une table il s'agit d'une jointure.

MySQL crée à chaque fois que l'on fait un SELECT (extraction de données) une pseudo-table, elle n'est pas gardée en mémoire.

mysql>SELECT 'Hello World !' ;
J'ai perdu ma montre
mysql>SELECT SYSDATE() 'Ma montre' ;
J'aime les mathématiques
mysql>SELECT POWER(2,14), COS(PI()), DEGREES(PI()), EXP(1) ;

POWER(2,14)	COS(PI())	DEGREES(PI())	EXP(1)
16384	-1	180	2.718281828459045

En résumé :

SELECT ref, title ← projection
FROM foo ← nom de la table
WHERE foo_id>1 AND status='publish' ← restriction sous ensemble des données de la table
ORDER BY date_crea ASC ← ordre ASC ou DESC classer les résultats
LIMIT 0,10 ← 0 : offset 10 le nombre d'éléments du tableau

Exercices (DQL)

Vous récupérez les fichier compagny.sql dans le dossier sql puis vous exécutez ce fichier à l'aide de la commande SOURCE dans votre terminal pour installer la base de données db_compagny, les tables et les données d'exemple.

mysql>SOURCE chemin_vers_compagny.sql ;

Exercice 1
Il existe une directive DISTINCT en SQL, aidez-vous de cette directive et afficher les différentes compagnies

Exercice 2
Sélectionnez tous les pilotes dont l'id est compris strictement entre 2 et 5 strictement.

Exercice 3
Trouver le nombre de vols le plus important de la table pilots.
Puis le nombre de vol minimum.
Et enfin les deux dans la même pseudo-table (table qui s'affiche dans la console). On utilisera la clause MAX et MIN dans la partie SELECT sur les colonnes concernées.

Exercice 4
Si vous faites la requête suivante vous obtenez le nom du premier champ de la table ainsi que le max du nombre de vols :

SELECT MAX(number_flight), name FROM pilots;

Donc cela ne marche pas pour trouver le nom du pilote ayant fait le plus grand nombre de vols, trouver la requête SQL pour obtenir le nom de ce pilote.

Exercice 5
On ajoutera les colonnes suivantes : bonus, type_plane, job_date dans la table Pilot. Écrire la commande DDL pour modifier la structure de la table (ALTER TABLE …) en optimisant l'espace mémoire pour les variables SQL.On notera que le bonus est un décimal ayant 6 chiffres en tout et deux après la virgule. Le type_plane sera une chaîne de caractères fixe 5 lettres, job_date un champ de type DATETIME.

Puis mettre à jour, cette table à l'aide des lignes suivantes :

UPDATE pilots SET bonus=100, type_plane='A380', job_date='2012-12-31 20:59:59' WHERE id>5 AND id<=7;

UPDATE pilots SET bonus=300, type_plane='A320', job_date='2010-12-31 15:10:59' WHERE pilot_id<=5;

UPDATE pilots SET bonus=500, type_plane='A340', job_date='2009-12-31 18:00:00' WHERE id=8;

Exercice 6
A l'aide de l'opérateur LIKE qui s'utilise avec la clause WHERE comme suit : WHERE col1 LIKE 'chaine' on utilisera les jokers suivants : **%** ou **_**, le premier joker indique n'importe quel nombre de caractères le deuxième exactement un caractère.
'%foo %' recherche un texte, une chaîne ayant le mot clef foo à l'intérieur.

Trouver tous les pilotes qui ont un « a » dans leur nom, tous les pilotes dont le nom est composé (signe -), tous les noms commençant par un « a » uniquement, tous les noms dont l'avant dernière lettre est un « l ».

Fonctions de groupe

Elles s'utilisent dans la clause SELECT sur une/des colonnes, elles permettent de regrouper des données. Si vous utilisez les fonctions de groupement avec une requête ne contenant pas de clause GROUP BY, cela revient à grouper toutes les lignes. Si vous l'utilisez avec WHERE cela fera la moyenne sur la restriction des lignes demandées.

AVG([DISTINCT] exp) moyenne
COUNT({*| DISTINCT] expr}) nombre de lignes → COUNT(*) ou COUNT(DISTINCT field)
MAX([DISTINCT] exp) max
MIN([DISTINCT] exp) min
SUM([DISTINCT] exp) somme
GROUP_CONCAT(exp) composition d'un nombre de valeurs, concaténation de valeurs de champ a, b, c, d

Exercice 7
On donnera la moyenne des heures de vol et des primes(bonus) des pilotes de la compagnie 'AF'

Exercice 8
On donnera le nombre de primes (bonus) distincts de la table pilots, et le nombre de compagnies distinctes.

Exercice 9
On donnera les noms des pilotes de la compagnie AF, sur une ligne.

GROUP BY, HAVING

La clause GROUP BY s'applique au niveau **surligné** de l'instruction SQL ci-dessous, les colonnes présentes dans le SELECT doivent apparaître dans le GROUP BY, seul les fonctions ou expression peuvent exister en plus dans le SELECT. On ne peut pas utiliser d'alias dans la clause GROUP BY.

SELECT col1, col2, ..., [fonctions de groupement]
[FROM tableName
[WHERE where_definition]
[**GROUP BY {col1, col2, ...}** [ASC | DESC], ...
[HAVING where_definition]
]

La clause WHERE permet d'exclure des lignes pour chaque groupement, ou rejeter des groupements entiers.

La clause GROUP BY liste des colonnes de groupement (les regroupes).

La clause HAVING permet de poser des conditions sur chaque groupement (sorte de filtre sur les données) elle ne s'utilise que sur une fonction d'agrégat (AVG, MAX, MIN, ...). Attention, il faut bien comprendre que la clause WHERE agit avant HAVING qui fait un filtre sur un résultat après que la clause WHERE ait fait son travail, logique en somme.

Exercice 10
On donnera la moyenne des heures de vol et des primes pour chaque compagnie

Exercice 11
Le nombre d'heures de vol le plus élevé pour chaque compagnie.

Exercice 12
Donnez le nombre de pilotes par type d'appareil et par compagnie.

Exercice 13
Combien d'avion distinct par compagnie

Exercice 14
Donner le nombre de pilotes des compagnies qui ont plus d'un pilote.

Exercice 15
On aimerait savoir quels sont les types d'avions en commun que la compagnie AF et AC exploitent. On peut voir le problème comme deux ensembles, l'ensemble des avions de la compagnie AC et l'ensemble des avions de la compagnie AF. On utilisera les opérateurs ensemblistes IN et AND :

(SELECT col …) AND col IN (SELECT …)

CLEF PRIMAIRE et SECONDAIRE

PRIMARY KEY clef primaire, elle peut être composite (plusieurs nom de colonnes) <=> UNIQUE NOT NULL INDEX
Rappelons qu'un INDEX est une structure supplémentaire créé par MySQL pour garder l'ordre des colonnes indexées.

FOREIGN KEY clef secondaire (éventuellement composite) elle se réfère à un INDEX ou clef primaire. Attention, on doit utiliser le moteur InnoDB, MyISAM n'est pas relationnel (pas de clef secondaire).
Elle permet de ne pas insérer de données qui n'auraient pas de sens dans les tables. Elle permet l'intégrité des données.
Pour créer une clef secondaire on doit :
- choisir la ou les colonnes **FOREIGN KEY**
- choisir dans l'autre table la ou les colonnes qui va/vont servir de référence(s), REFRENCES.
La clef secondaire ne peut s'ajouter dans la description d'une colonne, on met sa définition en général en dessous de la clef primaire.

Exercice 13
On va créer une clef secondaire sur la table posts. On se connecte à la base « blog » :

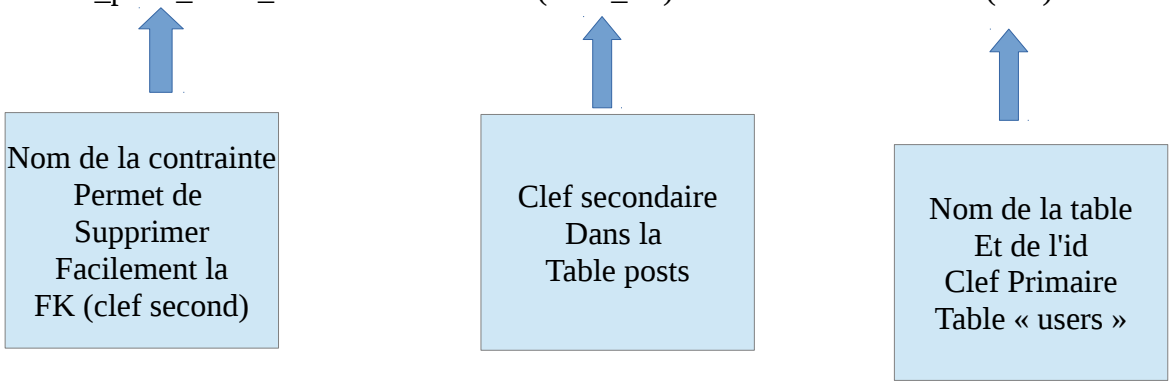
mysql>USE blog ;

On va maintenant ajouter une clef secondaire à la table posts pour relier cette table à la table users.
On rappelle la def de la table posts pour voir ce que l'on avait déjà fait :

mysql> show create table posts ;

```
CREATE TABLE `posts` (
  `id` INT UNSIGNED AUTO_INCREMENT,
  `user_id` INT(10) UNSIGNED,      ← PK même type que la clef primaire qui y fera référence.
  `title` VARCHAR(30) NOT NULL,
  `content` TEXT NOT NULL,
  `date_crea` DATETIME,
  `status` ENUM('publish', 'unpublish', 'draft', 'dash') NOT NULL DEFAULT 'publish',
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_posts_users_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
) ENGINE=InnoDB  AUTO_INCREMENT=1 ;
```

```
CONSTRAINT `fk_posts_users_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
```



La convention de nommage de la contrainte est la suivante : nom de la table de départ _ nom de la table d'arrivée _ id clef primaire de la table d'arrivée.

De manière générale : une clef secondaire se réfère toujours à une clef primaire d'une autre table.

Quelques commandes utiles :

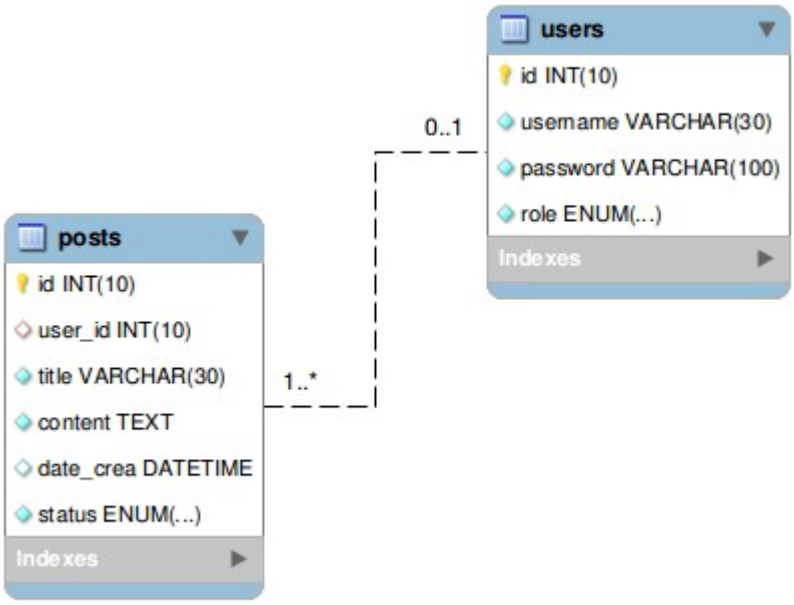
Supprime la contrainte, donc plus de relation entre la table users et posts.

ALTER TABLE `posts` DROP FOREIGN KEY `fk_posts_users_id` ;

Crée une contrainte FK dans la table posts.

ALTER TABLE posts ADD CONSTRAINT `fk_posts_users_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ;

Les tables sont maintenant physiquement reliés par des index.



Une fois les deux tables reliées vous ne pourrez plus ajouter un post dans la table posts avec un id d'auteur qui n'existe pas. Et vous ne pourrez pas également supprimer un user dans la table **users** qui a des posts dans la
table posts. Par contre vous pouvez supprimer un post ayant un user.

Un post à un unique user ou pas car le type n'est pas NOT NULL, et un user a 0 à N post(s).

Vérifiez en ligne de commande tout ce que l'on vient de dire.

Options des clefs étrangères :

On a deux types DELETE et UPDATE

ON DELETE [RESTRICT| NO ACTION | SET NULL | CASCADE] permet de déterminer le comportement de MySQL en cas de
suppression d'une référence

ON UPDATE [RESTRICT| NO ACTION | SET NULL | CASCADE] permet de déterminer le comportement de MySQL en cas de
modification d'une référence.

ON DELETE RESTRICT et NO ACTION comportement par défaut, ces deux options on le même effet dans MySQL uniquement mais pas dans les autres SGDB.

ON DELETE SET NULL dans ce cas NULL est substitué aux valeurs dont la référence est supprimée.

Exemple : si on a ON DELETE SET NULL sur la clef secondaire de la table « posts», et si on supprime une catégorie alors les posts qui avaient cette référence (id de la catégorie supprimé) ne sont pas supprimés et la valeur NULL est mise à la place de l'id de la catégorie dans la table « posts ».

ON DELETE CASCADE comportement plus risqué, plus voilent ! Elle supprime toutes les lignes. Par exemple si on supprime une catégorie dans la table « categories » reliée à la table posts (c'est dans cette table que l'on a la clef secondaire) alors tous les articles seront également supprimés.

Moins utiliser
UPDATE RESTRICT et NO ACTION : empêche la modification si elle casse la contrainte (comportement par défaut).

SET NULL : met NULL partout où la valeur modifiée était référencée.
CASCADE : modifie également la valeur là où elle est référencée.

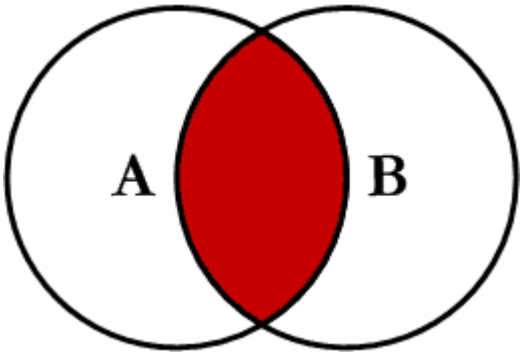
Exercice 14
Créez quelques users et ajouter dans la table posts des id de users.
Modifiez la structure de la table posts afin d'ajouter à la clef secondaire une option sur la suppression SET NULL.
Que constatez-vous si vous supprimez un users ayant des posts dans la table posts ?

Et maintenant testons le **ON DELETE CASCADE**.

Exercice 15
Modifiez la structure de la table et essayez de supprimer un user ayant plusieurs articles.

Jointure interne

SELECT t1.title, t2.title
FROM A as t1
INNER JOIN B as t2
ON t1.k=t2.k



Les jointures permettent d'extraire des données de plusieurs tables, dans le modèle relationnel SQL on décompose en tables (entités) les données, il est donc nécessaire de faire des jointures pour extraire des données se trouvant dans différentes tables.

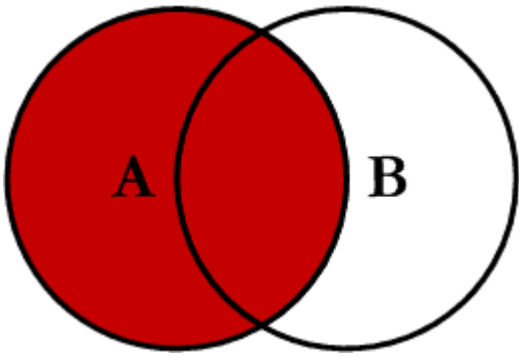
Principe :
Une jointure met en relation deux tables sur la base d'une clause de jointure (comparaison de colonne). Généralement, cette comparaison fait intervenir une clé étrangère d'une table avec une clé primaire d'une autre table. FK=PK, ils sont en particulier des indexes donc la requête s'effectue plus vite...

Le jointures interne (INNER JOIN)
Equijointure dans la clause de jointure on utilise « = »
Auto-jointure une équijointure sur la même table (on doit utiliser des alias pour distinguer les deux mêmes tables)

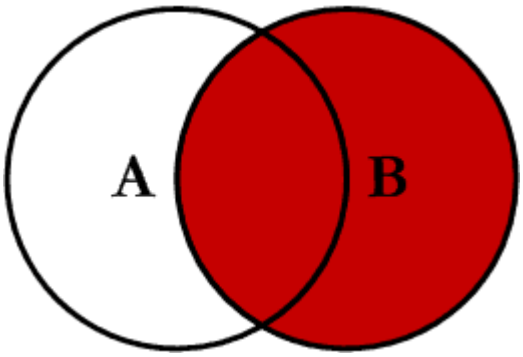
Dans une jointure interne seules les correspondances entre les deux tables sont extraites, on perd donc les données n'ayant pas de correspondances !

Exercice 16
Mettez dans la table users et posts des valeurs, des posts avec et sans users et réciproquement.

Extraire tous les posts et le nom de leur users dans une même pseudo-table.



Jointure externe



SELECT t1.title, t2.title
FROM A as t1
LEFT OUTER JOIN B as t2
ON t1.k=t2.k

SELECT t1.title, t2.title
FROM A as t1
RIGHT OUTER JOIN B as t2
ON t1.k=t2.k

Exercice 18
Extraire tous les posts avec le nom des users et sans.
Extraire tous les users ayant des posts et aucun.