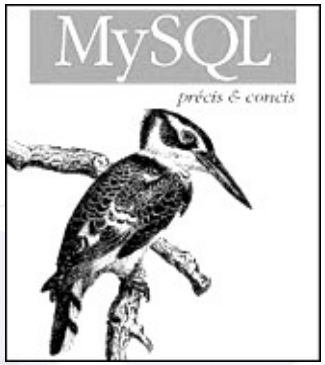
Informatique

Introduction au SGBD MySQL (et PHPMyAdmin)

Bruno Bernard SIMON
http://www.bbs-consultant.net



MySQL™

Plan :

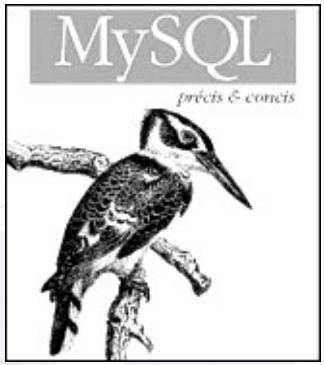
Principe des SGBD

Présentation de MySQL

Modèle relationnel

Structure MySQL

Syntaxe MySQL



Définition

Base de donées :

Collection de fichiers reliés par des pointeurs multiples, bien organisés, en vue de répondre efficacement à des demandes variées.

Collection d'informations organisée par une modélisation du monde réel.

SGBD

Système de Gestion de Bases de Données *(DBMS: Databases Management System)*

- Logiciel permettant le stockage de données persistantes et leur accès efficace. (Ullman)

SGBD

Système de Gestion de Bases de Données *(DBMS: Databases Management System)*

- Ensemble de logiciels systèmes permettant de stocker et d'interroger un ensemble de fichiers indépendants.
- Il est aussi un outil permettant de modéliser et de gérer des données. (G.Gardarin)

Historique

Années 60 : 1ère génération de SGBD : au début de l ’apparition des bases de données, le niveau conceptuel est très lié à la représentation des données sur les supports physiques (*) :

- modèle hiérarchique
- modèle réseau

(*) Analogie avec le vinyl et le CDA

Modèle relationnel

19701980 : 2ème génération plus indépendant des supports :

- modèle relationnel : SGBD-R

Débuts des années 80 : 3ème génération :

- modèle à objets.

Plan :

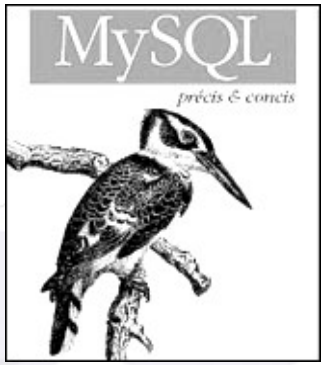
Principe des SGBD

Présentation de MySQL

Modèle relationnel

Structure MySQL

Syntaxe MySQL



SQL

MySQL dérive directement de SQL *(Structured Query Language)* qui est un langage de requête vers les bases de données exploitant le modèle relationnel.

MySQL en reprend la syntaxe mais n'en conserve pas toute la puissance, puisque de nombreuses fonctionnalités de SQL n'apparaissent pas dans MySQL : (sélections imbriquées, clés étrangères…

MySQL

MySQL est un SGBD-R (Système de Gestion de Bases de Données Relationnelles) client-serveur.

- Sa licence est libre ou propriétaire.
- Logiciel parmi les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle ou Microsoft SQL Server.

Logiciel libre ou propriétaire

C'est un logiciel libre développé sous double licence : 

- Licence libre GPL s'il est incorporé dans un produit libre ;
- licence payante s'il est utilisé dans un logiciel propriétaire.

Historique

MySQLAB est une Société suédoise, fondée par :

- David Axmark,
- Allan Larsson,
- Michael Widenius.

Historique

Le nom MySQL vient de leur habitude à préfixer par «My» une grande partie de leurs dossiers, bibliothèques et outils.

- Le nom du logo de MySQL (le dauphin) Sakila, a été proposé par Ambrose Twebaze, développeur du Swaziland.

Historique

Développé par Michael WIDENIUS.

- La première version de MySQL est apparue le 23 mai 1995.
- Depuis le logiciel libre a connu une popularité croissante et un développement international considérable.

Historique

MySQL fait partie des quatuors :

LAMP = Linux, Apache, MySQL, PHP.

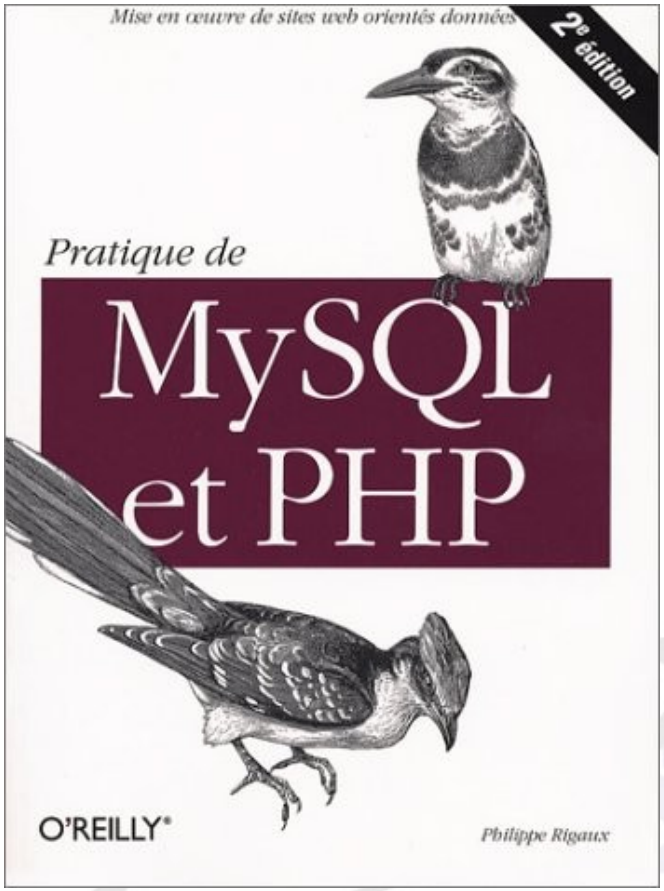
WAMP = Windows, Apache, MySQL, PHP.

MAMP = Mac, Apache, MySQL, PHP.

Plus de la moitié des sites web fonctionnent sous Apache, qui est le plus souvent utilisé conjointement avec PHP et MySQL.

Historique

Le couple MySQL/PHP est utilisé par une majorité de sites Web et proposé par la majorité des hébergeurs.



SGBD performant

Les utilisateurs de MySQL :

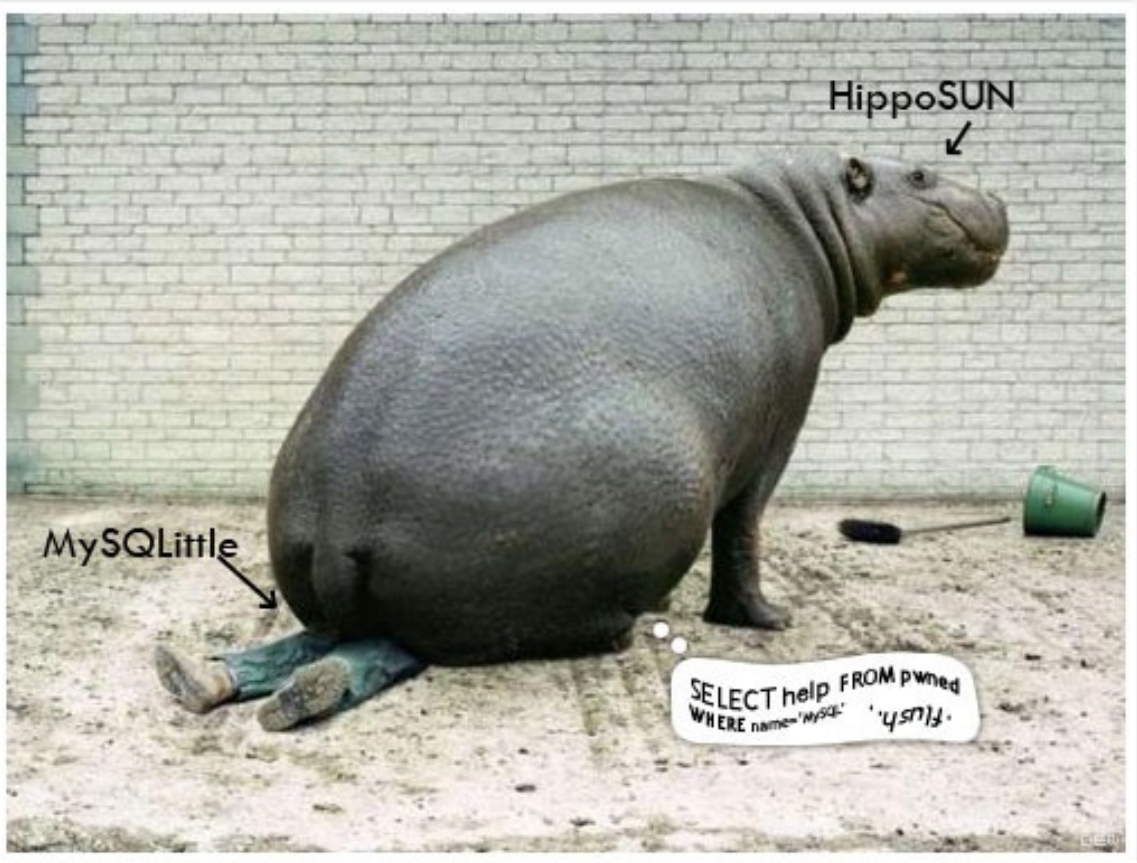
Wikipédia, Google, Yahoo!, YouTube, Adobe, Airbus, Alstom, Crédit agricole, Linden Lab (Second Life), RATP, URSSAF, AFP, Reuters, BBC News, Leader Price, Système U, Capgemini, Ernst & Young, Alcatel-Lucent, ...

Historique

MySQL AB a été acheté le 16 janvier 2008 par Sun Microsystems pour un milliard de dollars (1,000,000,000 $).

Le géant mondial du SGBD-R est l'américain Oracle Corporation, concurrent de Sun Microsystem et proposant un outil beaucoup plus puissant : « Oracle ».

SUN rachète MySQL



Historique

En avril 2009, Sun Microsystems a été acquis par Oracle Corp. (5,6 Md$) !

Cet OPA met entre les mains d'une même société les 2 concurrents Oracle et MySQL.

Ce rachat a failli 'être bloqué par l'Union Européenne en 2010, mais il a été finalement conclu.

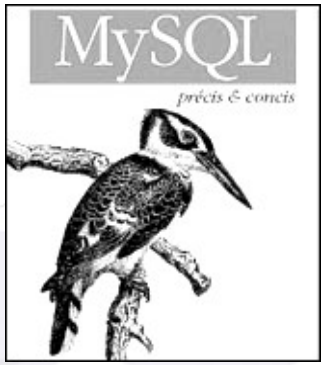
Maintien du logiciel libre

Depuis mai 2009, son créateur Michael WIDENIUS a créé MariaDB pour continuer son développement en tant que projet Open-Source.

En septembre 2013, Google abandonne MySQL pour MariaDB.

Plan :

Principe des SGBD
Présentation de MySQL
Modèle relationnel
Structure MySQL
Syntaxe MySQL



SGBD-R

Une base de données relationnelle est une collection de données structurées, elle comporte :

- des relations ou tables qui comportent :
- des attributs ou champs qui contiennent :
- des tuples ou enregistrements.

Relations ou tables

Une relation est une table comportant des attributs (appelées aussi champs) dont le nom et le type caractérisent le contenu qui sera inséré dans la table.

Les tuples que contiendra cette relation seront appelées ordinairement les enregistrements !

Commande CREATE TABLE ...()

```
CREATE TABLE Personne (
'nom' VARCHAR(40),
'prénom' VARCHAR(40),
'adresse' TINYTEXT,
'téléphone' DECIMAL(10,0))
```

Exemple de table

Une relation (table) nommée « Personnes »

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789

Algèbre relationelle

L'algèbre relationnelle regroupe toutes les opérations possibles sur les relations. Voici la liste des opérations possibles :

- Projection
- Sélection
- Jointure

Projection

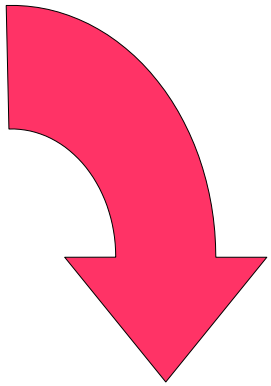
Projection : on ne sélectionne qu'un ou plusieurs champs d'une table :

Par exemple n'afficher que les champs « nom » et « prenom » de la table Personnes.

Projection

personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789



On projette la table Personnes sur les colonnes nom et prénom.

Syntaxe précisée au chapitre suivant.

nom	*prenom*
PAUL	Jacques
LAFRAISE	André
CARTIER	Jacques

Sélection

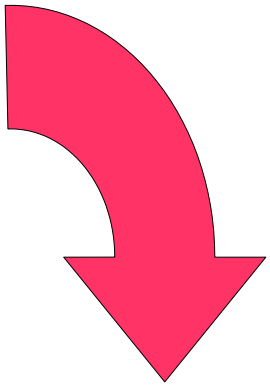
On sélectionne seulement une partie des enregistrements en fonction de critères de sélection qui portent sur les valeurs des enregistrements.

- Par exemple n'afficher que les lignes de la table Personnes qui vérifient la condition suivante : le prénom EST « Jacques »

Sélection

personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789



On sélectionne dans la table personnes les attributs voulus sur un critère de requête (filtre), ici le prénom jacques.

Syntaxe précisée au chapitre suivant.

nom	*prenom*
PAUL	Jacques
CARTIER	Jacques

Jointure

On fabrique une nouvelle relation à partir de 2 ou plusieurs autres en prenant comme pivot 1 ou plusieurs attributs.

Par exemple, on concatène la table du carnet d'adresse et celle des inscrits à la bibliothèque en fonction du nom de famille.

C'est typiquement du recoupement de fichiers.

Jointure

personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789

bibliotheque

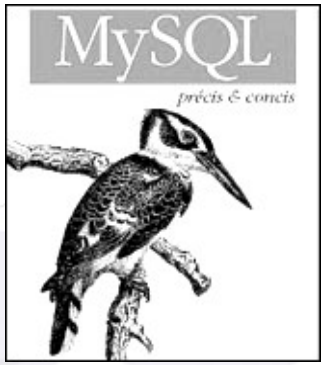
nom	*livre*
PAUL	Robinson
LAFRAMBOISE	Loup
CARTIER	Croc-Blanc

On joint les deux tables, grâce à la colonne *nom.*
Et on combine cette jointure à une projection sur les attributs *nom* et *livre*.

prenom	*Dernierlivre*
Jacques	Robinson
Lacques	Croc-blanc

Plan :

Principe des SGBD
Présentation de MySQL
Modèle relationnel
Structure MySQL
Syntaxe MySQL



Propriétes des attributs (champs)

Les propriétés des champs peuvent être :

- numérique : entier signé ou non ;
- numérique : nombre à virgule flottante ;
- alphanumérique : chaîne de caractères ;
- datation : date et heure ;
- énumération
    - une caractéristique parmi une liste prédéfinie.
- ensemble (une ou des caractéristiques
    - une ou plusieurs caractéristique parmi une liste prédéfinie.

Type numérique

Nom	Borne inférieure	Borne supérieure
TINYINT 2^8	-128	127
TINYINT UNSIGNED	0	255
SMALLINT 2^{16}	-32 768	32 767
SMALLINT UNSIGNED	0	65 535
MEDIUMINT 2^{24}	-8 388 608	8 388 607
MEDIUMINT UNSIGNED	0	16 777 215
INT* 2^{32}	-2 147 483 648	2 147 483 647
INT* UNSIGNED	0	4 294 967 295
BIGINT 2^{64}	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
BIGINT UNSIGNED	0	18 446 744 073 709 551 615

Grands nombres

Vous avez dit « milliard de milliards » ?

18 446 744 073 709 551 615
18 « quintillons » sur l'échelle courte,
18 trillions sur l'échelle longue !

http://www.alain.be/Boece/grands_nombres.html

Numérique à virgule flottante

Nom	Négatif : borne inférieure borne supérieure	Positif : borne inférieure borne supérieure
FLOAT	-3.402 823 466 E+38 -1.175 494 351 E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

Les flottants, ou nombres réels, sont des nombres à virgule. Contrairement aux entiers, leur domaine n'est pas continu : il est impossible de les représenter avec une précision absolue !

Notion de « rembourrage »

ZEROFILL : un attribut numérique peut être paramétré à ZEROFILL en option.

Cela signifie que le nombre contenu sera affiché avec le nombre maximal de chiffres, si besoin compélét par des 0.

- SMALLINT ZEROFILL de 1024 = 01024
- SMALLINT ZEROFILL de 512 = 00512
- SMALLINT ZEROFILL de 64 = 00064

Type alphanumérique

Nom	Longueur
CHAR(M)	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65 535 caractères.
MEDIUMTEXT	Longueur maximale de 16 777 215 caractères.
LONGTEXT	Longueur maximale de 4 294 967 295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

Type alphanumérique

Nom	Longueur
TINYBLOB	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65 535 caractères.
MEDIUMTEXT	Longueur maximale de 16 777 215 caractères.
LONGTEXT	Longueur maximale de 4 294 967 295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

Type BLOB (objet)

Les types BLOB sont substituables :

TINYTEXT	=>	TINYBLOB
TEXT	=>	BLOB
MEDIUMTEXT	=>	MEDIUMBLOB
LONGTEXT	=>	LONGBLOB.

Les types « TEXT » sont insensibles à la casse lors des tris et recherches.

Les types « BLOB » sont sensibles à la casse lors des tris et recherches.

Type datation

Nom	Description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais affiche les M premiers caractères de **TIMESTAMP**.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

Exemple : TIMESTAMP(8) = AAAAMMJJ

Le nombre de caractères affichable est de 2 à 14.

Type « énumération »

Un attribut « ENUM » peut contenir jusqu'à 65535 valeurs, insensibles à la casse.

Ces valeurs sont des chaînes uniquement.

A chaque valeur est associé un indice de 1 à n pour n valeurs définies. L'indice 0 renvoie une chaîne vide.

ENUM('pirate',flibustier','corsaire') NULL

Type « ensemble »

Un attribut de type « SET » peut prendre pour valeur la chaîne vide, NULL ou une chaîne contenant une liste de 64 valeurs au maximum.

SET('pirate',flibustier','corsaire') NULL

NOT NULL

Si l'on souhaite que certains attributs aient obligatoirement une valeur, il faut utiliser l'option « NOT NULL ».

Dans ce cas, un attribut se verra affecter :

- une chaîne vide s'il est du type alphanumérique ;
- la valeur zéro 0 s'il est de type numérique ;
- la date/heure nulle 0000-00-00/00:00:00

DEFAULT

Si l'on souhaite donner une valeur par défaut à un attribut, on utilise l'option « DEFAULT » et définir une valeur.

Lors de l'ajout d'un enregistrement cette valeur sera affectée à l'attribut si aucune valeur n'est donnée.

- Un attribut TEXT ou BLOB ne peut pas avoir de valeur par défaut !

UNIQUE

Si l'on souhaite interdire l'insertion d'un doublon dans un attribut, il faut utiliser l'option « UNIQUE ».

On peut, par exemple, interdire tout doublon de l'attribut « pseudo » et l'attribut « motdepasse » d'une relation :

```
UNIQUE(pseudo)
UNIQUE(motdepasse)
```

INDEX

Lors de la recherche d'informations (SELECT), MySQL parcourt toute la table correspondante (relation).

Dans le cas d'un grand nombre de lignes, cette recherche peut devenir très longue.

Les « index » pallient ce problème.

INDEX

Créer un index associé à un attribut ou à un ensemble d'attributs va créer une « liste ordonnée » des valeurs de ces attributs.

Les recherches et les tris se feront alors sur ces listes ou index.

Les algorithmes de recherche et de tri sur des ensembles ordonnées sont énormément plus rapides !

INDEX

Syntaxe :

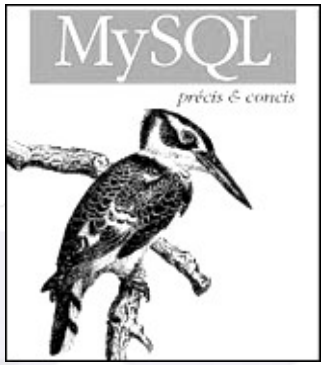
INDEX(nom)
INDEX(nom,(3))
INDEX(nom,prenom)

Limites :

Une relation supporte 16 index maximum ;
Un index porte sur 15 attributs au maximum ;
Un attribut NOT NULL ne peut être indexé ;
Un index ne peut dépasser 256 octets ;

Plan :

Principe des SGBD
Présentation de MySQL
Modèle relationnel
Structure MySQL
Syntaxe MySQL



Règles syntaxiques

Le point « . » est un caractère réservé utilisé comme séparateur entre le nom d'une base et celui d'une relation, entre le nom d'une relation et celui d'un attribut.

Exemple : SELECT base1.relation1.attribut5
Ou SELECT base1.table1.champ1

Création de table

```
CREATE TABLE  client (
nom VARCHAR(80) NOT NULL ,
prenom VARCHAR(80) NOT NULL ,
sexe ENUM("Homme",  "Femme") NOT
NULL ,
ddn DATE NULL ,
adresse BIGINT UNSIGNED NOT NULL
)
```

Création de table temporaire

```
CREATE [TEMPORARY] TABLE  panier [IF NOT EXISTS](
code VARCHAR(80) NOT NULL ,
produits VARCHAR(250) NOT NULL ,
montant MEDIUMINT UNSIGNED NOT NULL
)
```

Cette table sera détruite en fin de session, suite à la déconnexion du serveur MySQL.

Création de table non vide

```
CREATE TABLE test_2
  AS (SELECT *
        FROM test
        WHERE prenom='Jacques');
```

Cette table sera créée et renseignée à partir d'une sélection de la table test.

Modification de table

Bien sûr, il sera toujours possible de modifier la définition de la table (ALTER TABLE) pour modifier ou ajouter une rubrique :

```
ALTER TABLE  client CHANGE  adresse
adresse MEDIUMINT( 8 ) UNSIGNED NOT NULL
```

Mais il est indispensable de créer toutes ses tables sur le papier AVANT de les implanter sous MySQL afin de prévoir toutes les relations possibles.

Suppression de table

La commande « DROP TABLE » prend en paramètre le nom de la table à supprimer.

```
DROP TABLE 'test'
```

Toutes les données qu'elle contient sont supprimées et sa structure aussi.

Renommage de table

ALTER TABLE test RENAME client

Cela consiste à renommer la table, et donc le fichier qui la stocke.

Modification d'attribut

Pour changer le type :

```
ALTER TABLE test MODIFY nom VARCHAR(128)
```

Pour changer le nom ET le type :

```
ALTER TABLE test CHANGE nom patronyme VARCHAR(128)
```

Suppression d'attribut

ALTER TABLE 'test' DROP 'photo'

Attention, supprimer un attribut implique la suppression de toutes les valeurs de cet attribut pour tous les tuples de la relation.

Un attribut ne peut être détruit s'il a été déclaré en unicité jointe à un autre attribut.

Clef primaire sur attributs

```
CREATE TABLE Personne (
nom VARCHAR(40),
prenom VARCHAR(40),
pseudo VARCHAR(24),
passe VARCHAR(16),
telephone DECIMAL(10,0),
PRIMARY KEY (pseudo,passe)
)
```

Contrainte d'unicité d'un attribut

Changer la valeur par défaut :

ALTER TABLE test ALTER telephone SET DEFAULT ‘9999999999’

Supprimer la valeur par défaut :

ALTER TABLE test ALTER telephone DROP DEFAULT

Notion d'identifiant

Un enregistrement doit pouvoir être identifié par un identifiant numérique unique utilisable dans une variable de script, par exemple.

En fonction du nombre de tuples (enregistrements), on sélectionnera le type numérique idoine :

id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL

Notion de « clef primaire »

Une clef primaire peut être un identifiant numérique unique à incrémentation automatique.

ATTENTION : la numérotation des clés primaires, débute à 1 et pas à 0 !

id SMALLINT UNSIGNED PRIMARY KEY AUTO_INCREMENT

Clef primaire sur attributs

Une clé primaire peut être associée simultanément à plusieurs attributs.

Si au lieu de créer un identifiant numérique unique, on souhaite simplement interdire d'avoir des doublon sur le couple (nom,prenom) et d'en interdire la nullité, on va créer une clé primaire sur ce couple.

Suppresion de clef primaire

Comme une clé primaire est unique, il n'y a aucune ambiguïté lors de la suppression :

```
ALTER TABLE test DROP PRIMARY KEY
```

Contrainte NULL

On peut appliquer la contrainte « NULL » à un attribut.

Si la valeur est omise lors d'une insertion de tuple, le mot « NULL » sera forcé comme valeur de l'attribut *.

* Sauf si l'option « valeur par défaut » a été choisie

Création de tuple

Ajouter un enregistrement, c'est-à-dire créer un tuple revient à « ajouter une ligne dans la table ».

- Les valeurs omises sont forcées par la valeur de l'option « default ».
- Sans option « default », MySQL impose la valeur « NULL » si la contrainte NULL a été retenue.
- Sinon, MySQL va forcer la valeur selon le type...

Création de tuple

MySQL force la valeur lors de l'insertion selon le type de l'attribut :

- 0 pour un attribut numérique ;
- “” (chaîne vide) pour un attribut alpha ;
- 0000-00-00 pour une type date ;
- 00:00:00 pour un type horaire ;
- 00000000000000 pour un tampon temporel.

Tampon temporel = TIMESTAMP

Création & auto-incrémentation

```
INSERT INTO users ( id, pseudo)
VALUES ('1', 'bob', );
```

Si le champs id est déclaré en auto-increment, il suffit d'écrire et le champ 'id' est automatiquement renseigné :

```
INSERT INTO users ( id, pseudo)
VALUES ('bob');
```

Insertion étendue

INSERT INTO test (nom,prenom)
VALUES (‘DIAZ’,’Bartolomeo’)

Si le tuple à insérer compromet l'unicité (UNIQUE, PRIMARY KEY) d'un tuple existant, MySQL refusera l'insertion et génèrera un message d'erreur :

« DUPLICATE ENTRY »

REPLACE

```
REPLACE INTO test (nom,prenom)
VALUES ('DIAZ','Bartolomeo')
```

REPLACE est un synonyme de INSERT, mais permet de modifier les attributs à contraintes d'unicité (UNIQUE, PRIMARY KEY).

Insertion complète

Dans le cas où l'on souhaite procéder à l'insertion de plusieurs enregistrements, il faut procéder à une insertion dite « complète » :

INSERT INTO relation VALUES (liste des valeurs), (liste d'autres valeurs), (liste d'encore d'autres valeurs), …

… où « liste des valeurs » donne une VALEUR pour CHAQUE attribut exhaustivement et dans l'ordre des attributs !

Insertion complète et étendue

Il est possible de procéder à une insertion complète et étendue simultanément.

C'est-à-dire en donnant la liste des attributs et leurs valeurs correspondantes :

```
INSERT INTO test(nom,prenom) VALUES
(CARTIER,Jacques), (LAFRAISE,Paul),
(DIAZ,Bartolomeo), (RALEIGH,Walter)
```

Modification de tuple(s)

Pour modifier un ou plusieurs tuples d'une relation, il faut désigner quels attributs à modifier, déclarer leurs valeurs (SET) et filtrer la relation sur un critère de sélection (WHERE) :

```
UPDATE [LOW_PRORITY ] test
SET ddn='14911223'
WHERE nom='CARTIER'
```

Modification de tuple(s)

L'exemple précédent modifie la date de naissance de Jacques CARTIER. Mais il est possibles de modifier dans la même requête plusieurs attributs !

LOW_PRORITY est une option un peu spéciale qui permet de n'appliquer la ou les modification(s) qu'une fois que plus personne n'est en train de lire dans la relation.

Suppression de tuple(s)

Attention, la suppression est définitive !

DELETE [LOW_PRIORITY] FROM test
[WHERE nom= 'CARTIER'] [LIMIT 1]

- WHERE permet de filtrer la suppression et LIMIT est une option qui limite dans cet exemple la suppression à un seul tuple.

DELETE [LOW_PRIORITY] FROM test

- Cette commande efface TOUS les tuples !

Optimisation

Après la suppression de grandes parties d'une table contenant des index, les index des tuples supprimés sont conservés, rallongeant d'autant les sélections.

Pour supprimer ces index obsolètes et vider les « trous », il faut l'optimiser.

OPTIMIZE TABLE *Personnes*

Projection de tuples

personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789

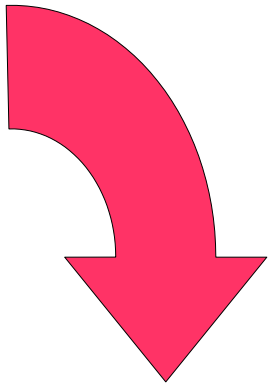
SELECT *nom, prenom*
FROM *personnes*

nom	*prenom*
PAUL	Jacques
LAFRAISE	André
CARTIER	Jacques

Sélection de tuples

Personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789



```
SELECT nom, prenom
FROM Personnes
WHERE prenom = 'Jacques'
```

nom	*prenom*
PAUL	Jacques
CARTIER	Jacques

Syntaxe de SELECT

```
SELECT [ DISTINCT ] attributs
[ INTO OUTFILE fichier ]
[ FROM relation ]
[ WHERE condition ]
[ ORDER BY attributs ]
[ LIMIT [d,] n ]
[ GROUP BY attributs [ ASC | DESC ] ]
[ HAVING condition ]
```

Syntaxe de SELECT

Nom	Description
SELECT	Spécifie les attributs dont on souhaite connaître les valeurs.
DISTINCT	Permet d'ignorer les doublons de ligne de résultat.
INTO OUTFILE	Spécifie le fichier sur lequel effectuer la sélection.
FROM	Spécifie le ou les relations sur lesquelles effectuer la sélection.
WHERE	Définie le ou les critères de sélection sur des attributs.
GROUP BY	Permet de grouper les lignes de résultats selon un ou des attributs.
HAVING	Définie un ou des critères de sélection sur des ensembles de valeurs d'attributs après groupement.
ORDER BY	Permet de définir l'ordre (**ASC**endant par défaut ou **DESC**endant) dans l'envoi des résultats.
LIMIT	Permet de limiter le nombre de lignes du résultats

Clause DISTINCT

La clause DISTINCT élimine les doublons lors d'une sélection.

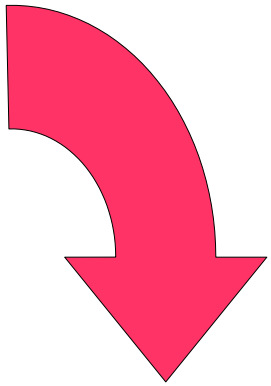
Elle permet donc d'obtenir la liste de toutes les valeurs différentes d'un attribut.

Elle s'emploi de préférence avec l'option ORDER BY.

Clause DISTINCT

Personnes

nom	*prenom*	*adresse*	*telephone*
PAUL	Jacques	7 allée des mouettes	0258941236
LAFRAISE	André	32 avenue Surcouf	0526389152
CARTIER	Jacques	8 rue de la mer	0123456789



SELECT DISTINCT *prenom*
FROM *Personnes*
ORDER BY *prenom*

prenom
André
Jacques

Clause ORDER BY

La clause ORDER BY *attribut* permet de classer les tuples sélectionnés dans l'ordre ascendant (ASC) ou descendant (DESC) sur la valeur d'un tuple.

Cette clause est particulièrement utile pour produire des listes d'enregistrements triés en même temps qu'ils sont sélectionnés.

Clause LIMIT

La clause LIMIT n permet limiter la sélection à n tuples, où n est un entier qui précise le nombre de tuples à retourner.

Cette clause est indispensable pour limiter les listes de réponses sur un moteur de recherche ou les listes des articles d'un blog ou d'un forum.

Clause LIMIT

La clause LIMIT d,n permet limiter la sélection à n tuples, mais cette fois à partir d'un tuple d donné :

- d est le n° du premier tuple à retourner ;
- n est le nombre de tuples à retourner.

Clause GROUP BY

La clause GROUP BY *attribut* se combine avec la fonction COUNT() et permet d'effectuer des calculs sur chaque itération de la valeur d'un tuple...

Cette clause peut être utilisée, par exemple, pour compter dans une table le nombre de prénoms différents, sachant qu'il y a des prénoms identiques obligatoirement.

Clause GROUP BY

```
SELECT prenom, COUNT(prenom )
FROM test
GROUP BY prenom
```

Cette requête va créer une table où chaque ligne contiendra dans la première colonne tous les prénoms différents et dans la seconde, le total des occurrences de ces prénoms !

MIN(x), MAX(x)

Afficher le nom et l'âge du plus jeune client d'une table 'clients' :

```
SELECT nom, MIN( age )
FROM  clients
```

Afficher le nom et l'âge du plus vieux client d'une table 'clients' :

```
SELECT nom, MAX( age )
FROM  clients
```

Clause HAVING

La clause HAVING ne s'utilise que suite à la clause GROUP BY, pour adjoindre une "sous-sélection" sous forme de calculs ou autres.

Elle doit obligatoirement être placée après la clause GROUP BY, et avant toute éventuelle clause ORDER BY.

L'utilisation de HAVING avec des fonctions comme SUM(), AVG(), MAX() est fréquente.

JOINTURES

Les jointures simples

Les jointures complexes :

les jointures internes

les jointures externes

Jointure simple

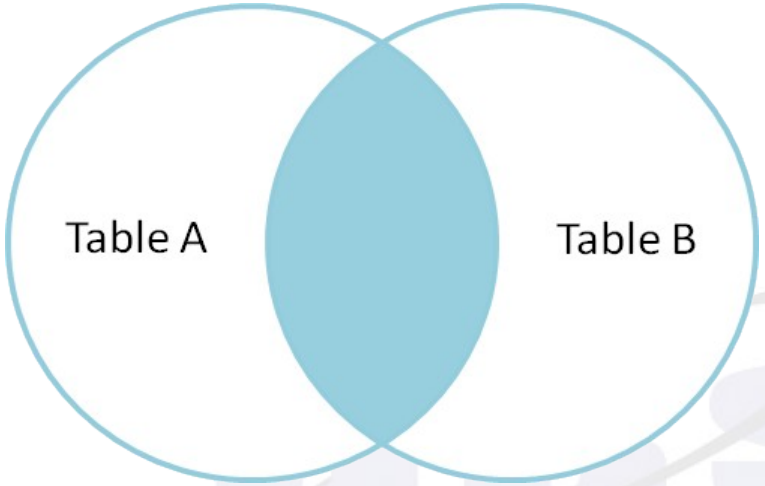
La jointure simple liste les éléments communs à deux tables.

La jointure simple ne requiert aucune commande spéciale de jointure : elle n'utilise que la clause WHERE et l'opérateur égal = .

Jointure simple

Une jointure simple est retourne l'intersection des tables A et B selon la théorie ensembliste :

Seuls les tuples présents dans A et B sont projetés !



Jointure simple

personnes

id	nom	mel
1	Jacques	jak@sfr.fr
2	Paul	paul@laposte.fr
3	Marie	m7@gmail.com
4	Julie	julie@hotmail.com
5	Thomas	thomas@nordnnt.fr

livres

ids	titre	id
99	Le maître de Garamod	1
100	Le jardin des Finzi Contini	3
101	La mort d'un apiculteur	2
102	Hôtel Lutetia	4
103	Narcisse & Goldmund	3

Jointure simple

```
SELECT *
FROM personnes, livres
WHERE personnes.id = livres.id;
```

id	nom	mel	ids	titre	id
1	Jacques	jak@sfr.fr	99	Le maître de Garamond	1
2	Paul	paul@laposte.fr	101	L'un mort d'un apiculteur	2
3	Marie	m7@gmail.com	100	Le jardin des Finz-Contini	3
3	Marie	m7@gmail.com	103	Narcisse et Goldmund	3
4	Julie	julie@hotmail.com	102	Hotel Lutetia	4

Sans clause WHERE, la jointure n'a aucun sens !

Jointures complexes

Les jointures complexes sont de type interne (INNER) ou externe (OUTER) :

- INNER : Une jointure interne fonctionne exactement de la même manière qu'une jointure simple.
- OUTER : Les jointures externes permettent de dépasser les limites de la jointure interne.

Jointures complexes

Les jointures complexes contrairement aux jointures simples disposent de leurs propres commandes :

- INNER JOIN
- OUTER LEFT JOIN
- OUTER RIGHT JOIN

Jointure (complexe) interne

Une jointure interne INNER JOIN fonctionne exactement de la même manière qu'une jointure simple !

Seules les commandes sont différentes.

Il est conseillé d'utiliser les commandes de jointure complexes pour les seules raisons de lisibilité.

Jointure (complexe) interne

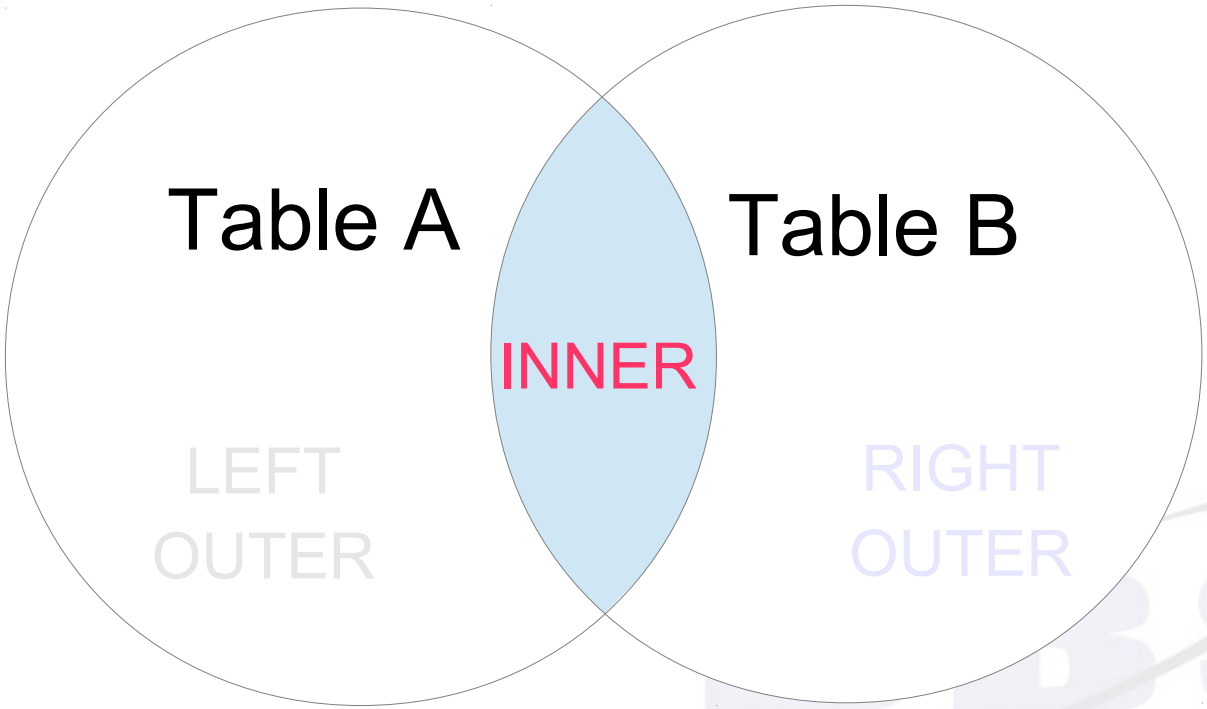
Pour remédier aux limites de la jointure interne INNER JOIN, il existe la syntaxe LEFT OUTER JOIN * :

Elle inclue tous les enregistrements de la première table même s'ils n'ont pas de correspondance dans la seconde table.*

* Dénommé aussi « jointure externe gauche »

INNER JOIN

Jointure « interne »



Jointure (complexe) interne

```
SELECT *
FROM personnes
INNER JOIN livres
ON personnes.id = livres.id
```

id	nom	mel	ids	titre	id
1	Jacques	jak@sfr.fr	99	Le maître de Garamond	1
2	Paul	paul@laposte.fr	101	L'un mort d'un apiculteur	2
3	Marie	m7@gmail.com	100	Le jardin des Finz-Contini	3
3	Marie	m7@gmail.com	103	Narcisse et Goldmund	3
4	Julie	julie@hotmail.com	102	Hotel Lutetia	4

Jointure complexe interne vs simple

```
SELECT *
FROM personnes
INNER JOIN livres
ON personnes.id = livres.id
```

Toujours préférer la syntaxe INNER JOIN pour des raisons de clarté de commande.

Rappel de la jointure simple :

```
SELECT *
FROM personnes, livres
WHERE personnes.id = livres.id;
```

Ici la notion de jointure n'apparaît pas de manière évidente !

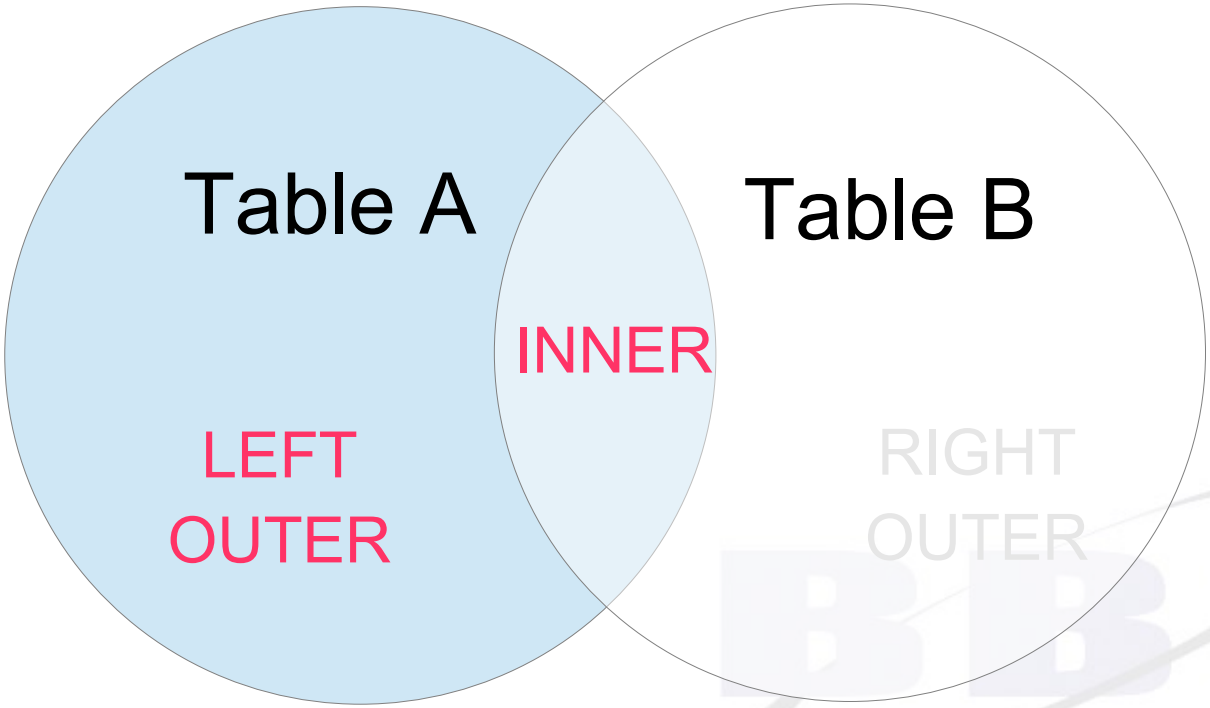
Jointure (complexe) externe gauche

Une jointure externe dépasse donc les limites de la jointure interne en ramenant tous les éléments de la table A !

Dans ce cas précis, l'attribut non renseigné prendra la valeur NULL.

LEFT OUTER JOIN

Jointure « externe gauche »



Jointure (complexe) externe gauche

personnes

id	nom	mel
1	Jacques	jak@sfr.fr
2	Paul	paul@laposte.fr
3	Marie	m7@gmail.com
4	Julie	julie@hotmail.com
5	Thomas	thomas@nordnet.fr

livres

ids	titre	id
99	Le maître de Garamod	1
100	Le jardin des Finzi Contini	3
101	La mort d'un apiculteur	2
102	Hôtel Lutetia	4
103	Narcisse & Goldmund	3

Jointure (complexe) externe gauche

```
SELECT *
FROM personnes
LEFT OUTER JOIN livres
ON personnes.id = livres.id
```

id	nom	mel	ids	titre	id
1	Jacques	jak@sfr.fr	99	Le maître de Garamond	1
2	Paul	paul@laposte.fr	101	L'un mort d'un apiculteur	2
3	Marie	m7@gmail.com	100	Le jardin des Finz-Contini	3
3	Marie	m7@gmail.com	103	Narcisse et Goldmund	3
4	Julie	julie@hotmail.com	102	Hotel Lutetia	4
5	Thomas	tom@nordnet.fr	*NULL*	*NULL*	*NULL*

Jointure (complexe) externe droite

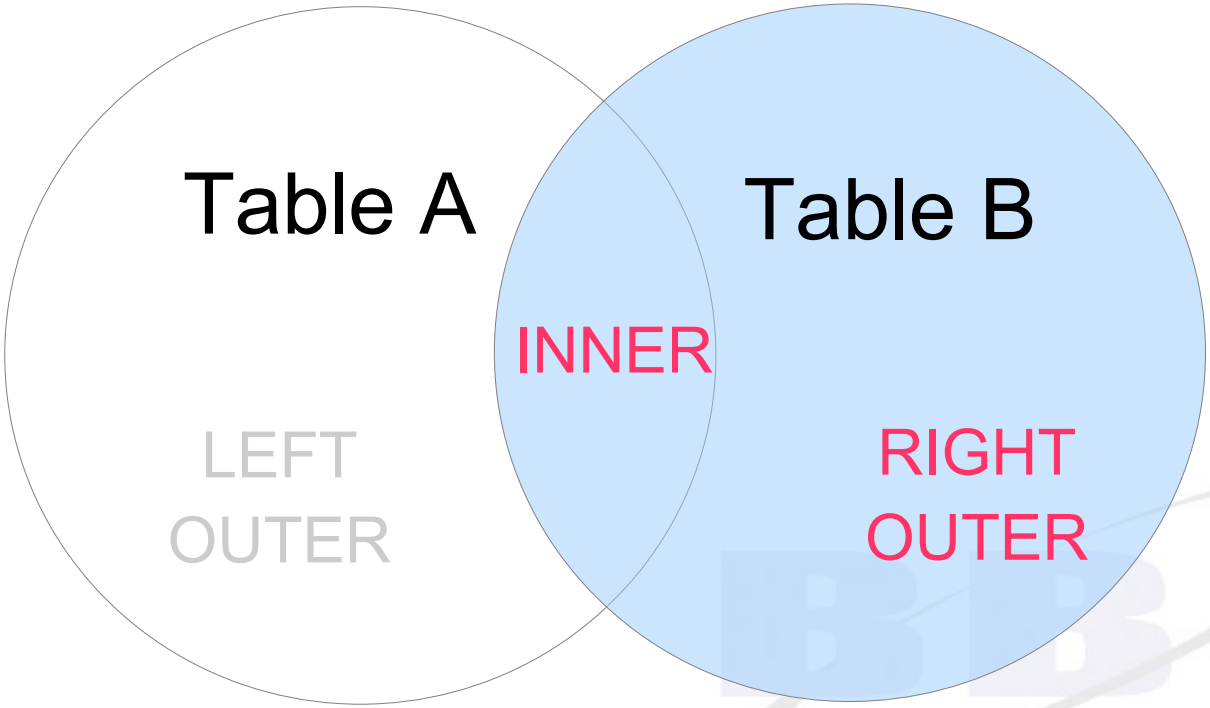
En sus de la jointure externe « gauche », LEFT OUTER JOIN, il existe la syntaxe RIGHT OUTER JOIN * :

Elle inclue tous les enregistrements de la DEUXIÈME table même s'ils n'ont pas de correspondance dans la PREMIÈRE table.

* Dénommé aussi « jointure externe droite »

RIGHT OUTER JOIN

Jointure « externe droite »



Jointure (complexe) externe droite

personnes

id	nom	mel
1	Jacques	jak@sfr.fr
2	Paul	paul@laposte.fr
3	Marie	m7@gmail.com
4	Julie	julie@hotmail.com
5	Thomas	thomas@nordnet.fr

livres

ids	titre	id
99	Le maître de Garamod	1
100	Le jardin des Finzi Contini	3
101	La mort d'un apiculteur	2
102	Hôtel Lutetia	4
103	Narcisse & Goldmund	3
104	Le meilleur des Mondes	6

Jointure (complexe) externe droite

```
SELECT *
FROM personnes
RIGHT OUTER JOIN livres
ON personnes.id = livres.id
```

id	nom	mel	ids	titre	id
1	Jacques	jak@sfr.fr	99	Le maître de Garamond	1
3	Marie	m7@gmail.com	100	Le jardin des Finz-Contini	3
2	Paul	paul@laposte.fr	101	L'un mort d'un apiculteur	2
4	Julie	julie@hotmail.com	102	Hotel Lutetia	4
3	Marie	m7@gmail.com	103	Narcisse et Goldmund	3
NULL	*NULL*	*NULL*	*104*	Le meilleur des mondes	6

Jointure (complexe) externe complète

Pour remédier aux limites de LEFT OUTER et RIGHT OUTER JOIN, il existe la syntaxe FULL OUTER JOIN dans le langage SQL...

… mais non supporté par MySQL !

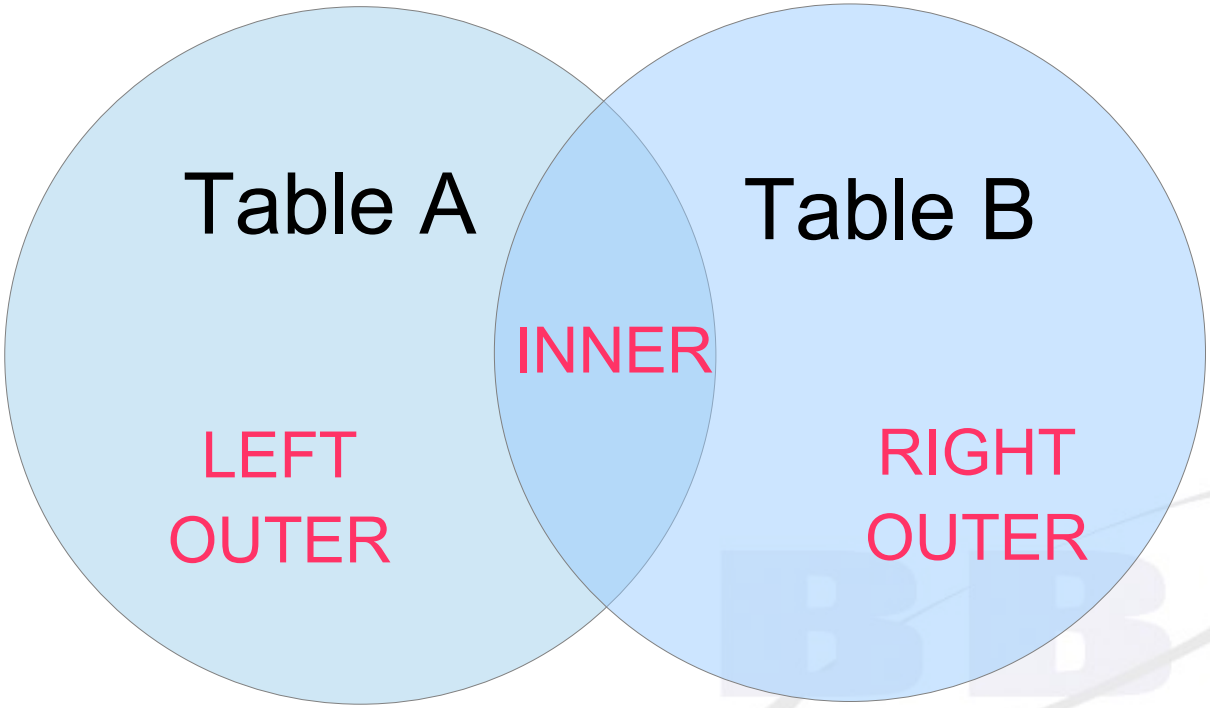
UNION MySQL

Pour réaliser sous MySQL une jointure externe complète, il faut utiliser la clause UNION appliquée à une jointure gauche et une jointure droite.

Une jointure externe complète peut en effet être vue comme l'union d'une jointure gauche et d'une joiture droite !

UNION

Jointure « externe complète »



Jointure (complexe) externe complète

```
(SELECT *
FROM personnes
LEFT OUTER JOIN livres
ON personnes.id = livres.id)
UNION
(SELECT *
FROM personnes
RIGHT OUTER JOIN livres
ON personnes.id = livres.id)
```

Jointure (complexe) externe complète

id	nom	mel	ids	titre	id
1	Jacques	jak@sfr.fr	99	Le maître de Garamond	1
2	Paul	paul@laposte.fr	101	L'un mort d'un apiculteur	2
3	Marie	m7@gmail.com	100	Le jardin des Finz-Contini	3
3	Marie	m7@gmail.com	103	Narcisse et Goldmund	3
4	Julie	julie@hotmail.com	102	Hotel Lutetia	4
5	Thomas	tom@nordnet.fr	*NULL*	*NULL*	*NULL*
NULL	*NULL*	*NULL*	*104*	Le meilleur des mondes	6

Jointures & ambiguïté

Attention à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des attributs de même nom :

```
SELECT *
FROM personnes,
LEFT OUTER JOIN livres
ON personnes.id = livres.id
```

USING ou ON

La syntaxe « USING » permet de lister les attributs servant de pivot.

Ces attributs doivent porter le même nom dans chacune des tables devant être concaténées.

Si les attributs pivots ne portent pas le même nom, il faut utiliser la syntaxe « ON ».

JOINTURES

Travaux pratiques sur les jointures

INNER JOIN

post_id	post_title	category_id
1	Bienvenue	1
2	Settimeout Javascript	2
3	Sprites CSS	3
4	Firebug	2
5	Google Apps	NULL
6	Colorisation Photoshop	3
7	Base photographie	5
8	Jointure SQL	4
9	Article top secret	NULL

Tables des articles

category_id	category_name
1	News
2	Javascript
3	Graphisme
4	SQL
5	Photographie
6	PHP

Table des catégories

```
SELECT *
FROM post AS p, category AS c
WHERE p.category_id = c.category_id;
```

post_id	post_title	category_id	category_id	category_name
1	Bienvenue	1	1	News
2	Settimeout Javascript	2	2	Javascript
4	Firebug	2	2	Javascript
3	Sprites CSS	3	3	Graphisme
6	Colorisation Photoshop	3	3	Graphisme
8	Jointure SQL	4	4	SQL
7	Base photographie	5	5	Photographie

http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql

INNER JOIN

```
SELECT client.id, nom, prenom,
commande.ref, date, paiement
FROM client
INNER JOIN commande
ON client.id = commande.client
WHERE client.id = "10"
```

LEFT OUTER JOIN

post_id	post_title	category_id
1	Bienvenue	1
2	Settimeout Javascript	2
3	Sprites CSS	3
4	Firebug	2
5	Google Apps	NULL
6	Colorisation Photoshop	3
7	Base photographie	5
8	Jointure SQL	4
9	Article top secret	NULL

Tables des articles

category_id	category_name
1	News
2	Javascript
3	Graphisme
4	SQL
5	Photographie
6	PHP

Table des catégories

```
SELECT *
FROM post AS p
LEFT JOIN category AS c ON p.category_id = c.category_id;
```

post_id	post_title	category_id	category_id	category_name
1	Bienvenue	1	1	News
2	Settimeout Javascript	2	2	Javascript
3	Sprites CSS	3	3	Graphisme
4	Firebug	2	2	Javascript
5	Google Apps	NULL	NULL	NULL
6	Colorisation Photoshop	3	3	Graphisme
7	Base photographie	5	5	Photographie
8	Jointure SQL	4	4	SQL
9	Article top secret	NULL	NULL	NULL

http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql

RIGHT OUTER JOIN

post_id	post_title	category_id
1	Bienvenue	1
2	Settimeout Javascript	2
3	Sprites CSS	3
4	Firebug	2
5	Google Apps	NULL
6	Colorisation Photoshop	3
7	Base photographie	5
8	Jointure SQL	4
9	Article top secret	NULL

Tables des articles

category_id	category_name
1	News
2	Javascript
3	Graphisme
4	SQL
5	Photographie
6	PHP

Table des catégories

```
SELECT *
FROM post AS p
RIGHT JOIN category AS c ON p.category_id = c.category_id;
```

post_id	post_title	category_id	category_id	category_name
1	Bienvenue	1	1	News
2	Settimeout Javascript	2	2	Javascript
4	Firebug	2	2	Javascript
3	Sprites CSS	3	3	Graphisme
6	Colorisation Photoshop	3	3	Graphisme
8	Jointure SQL	4	4	SQL
7	Base photographie	5	5	Photographie
NULL	NULL	NULL	6	PHP

http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql

FULL OUTER JOIN

post_id	post_title	category_id
1	Bienvenue	1
2	Settimeout Javascript	2
3	Sprites CSS	3
4	Firebug	2
5	Google Apps	NULL
6	Colorisation Photoshop	3
7	Base photographie	5
8	Jointure SQL	4
9	Article top secret	NULL

Tables des articles

category_id	category_name
1	News
2	Javascript
3	Graphisme
4	SQL
5	Photographie
6	PHP

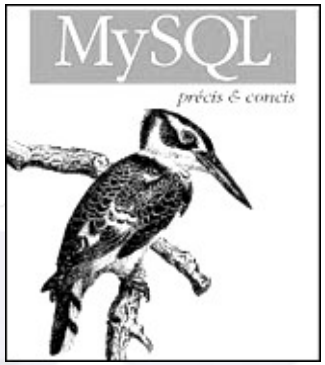
Table des catégories

Résultat

post_id	post_title	category_id	category_id	category_name
1	Bienvenue	1	1	News
2	Settimeout Javascript	2	2	Javascript
3	Sprites CSS	3	3	Graphisme
4	Firebug	2	2	Javascript
5	Google Apps	NULL	NULL	NULL
6	Colorisation Photoshop	3	3	Graphisme
7	Base photographie	5	5	Photographie
8	Jointure SQL	4	4	SQL
9	Article top secret	NULL	NULL	NULL
NULL	NULL	NULL	6	PHP

http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql

Plan :



Principe des SGBD
Présentation de MySQL
Modèle relationnel
Structure MySQL
Syntaxe MySQL
Fonctions MySQL

Règle de base

Les fonctions MySQL utilisent :

- les parenthèses ( )
- les opérateurs arithmétiques (+, -, *, /, %)
- les opérateurs booléens AND, OR, NOT, BETWEEN, IN qui retournent 0 (faux) ou 1 (vrai)
- les opérateurs relationnels (<, <=, =, >, >=, <>)
- les opérateurs binaires (<, << , >, >>, |, &)

Les opérateurs et les fonctions peuvent êtres composés entre eux pour donner des expressions très complexes.

Opérateurs relationnels

Égalité :

```
SELECT nom FROM produits
WHERE tva = 19.6
```

Supérieur / inférieur à / différent de :

```
SELECT nom FROM produits
WHERE prix > 99 AND <= 201
```

Opérateurs booléens

Fourchette booléenne BETWEEN :

```
SELECT nom,prenom FROM clients
WHERE age BETWEEN 18 AND 25
```

Opérateurs booléens

Opérateur booléen IN :

```
SELECT modele FROM chemises
WHERE couleur IN (‘rouge’, ‘blanc’, ’noir’)
```

Opérateur booléen NOT IN :

```
SELECT modele
FROM chemises
WHERE couleur NOT IN (‘rose’, ‘violet’)
```

Opérateurs de chaîne

L'opérateur LIKE permet de comparer deux chaînes :

```
SELECT nom
FROM test
WHERE prenom LIKE 'Jacques'
```

Cette commande permet de sélectionner les tuples dont l'attribut 'nom' est par 'Jacques'.

Opérateurs de chaîne

L'opérateur LIKE accepte 2 paramètres de troncature, les jokers : % et _ :

- Le joker % signifie : « 0 ou plusieurs caractères ».
- Le joker _ signifie : « 1 seul caractère, n'importe lequel ».

Joker (troncature) %

L'opérateur LIKE et son joker % :

```
SELECT nom,prenom FROM test
WHERE prenom LIKE 'Jac%'
```

Cette commande permet de sélectionner les tuples dont l'attribut 'prenom' commence par 'Jac'.

Joker (troncature) _

L'opérateur LIKE et son joker _ :

```
SELECT nom,prenom FROM test
WHERE nom LIKE 'DUPON_'
```

Cette commande permet de sélectionner les tuples dont l'attribut 'nom' EST « DUPOND » ou « DUPONT »

Opérateurs de chaîne

L'échappement des caractères spéciaux, à l'aide de la barre oblique inversée \ est obligatoire, comme en PHP :

```
SELECT ref,libelle
FROM produit
WHERE ref LIKE '\_DVD'
```

Cette commande permet de sélectionner les tuples dont l'attribut 'ref' commence par '_DVD'.

Fonctions de calcul

Fonction	Description
ABS(x)	Valeur absolue de X
SIGN(x)	Signe de X, retourne -1, 0 ou 1
FLOOR(x)	Arrondi à l'entier inférieur
CEILING(x)	Arrondi à l'entier supérieur
ROUND(x)	Arrondi à l'entier le plus proche
EXP(x), LOG(x), SIN(x), COS(x), TAN(x), PI()	Fonctions mathématiques de base
POW(x,y)	Retourne X à la puissance Y
RAND(), RAND(x)	Retourne un nombre aléatoire entre 0 et 1.0 Si x est spécifié, entre 0 et X
TRUNCATE(x,y)	Tronque le nombre X à la Yème décimale

Fonctions de chaînes

Fonction	Description
TRIM(x)	Supprime les espaces de début et de fin de chaîne.
LOWER(x)	Converti en minuscules.
UPPER(x)	Converti en majuscules.
LONGUEUR(x)	Retourne la taille de la chaîne.
LOCATE(x,y)	Retourne la position de la dernière occurrence de x dans y. Retourne 0 si x n'est pas trouvé dans y.
CONCAT(x,y,…)	Concatène ses arguments.
SUBSTRING(s,i,n)	Retourne les n derniers caractères de s en commençant à partir de la position i.
SOUNDEX(x)	Retourne une représentation phonétique de x.

Fonction d'horodatage

Fonction	Description
NOW()	Retourne la date et heure du jour.
TO_DAYS(x)	Conversion de la date X en nombre de jours depuis le 1er janvier 1970.
DAYOFWEEK(x)	Retourne le jour de la semaine de la date x sous la forme d'un index qui commence à 1 (1=dimanche, 2=lundi…)
DAYOFMONTH(x)	Retourne le quantième du mois (entre 1 et 31).
DAYOFYEAR(x)	Retourne le quantième de l'année (1 - 366).
SECOND(x), MINUTE(x), HOUR(x), MONTH(x), YEAR(x), WEEK(x)	Retournent respectivement les secondes, minutes, heures, mois, année et semaine de la date.

Fonction NOW()

La fonction NOW() retourne la date du jour au format MySQL :

2011-04-21 12:40:27

```
SELECT titre,chapo,parution,NOW()
FROM articles
```

Fonction TO_DAYS()

Fonction un peu particulière, TO_DAYS() retourne le nombre de jours entre la date passée en argument et le 01/01/1970 !

```
SELECT titre,chapo
FROM articles
WHERE (TO_DAYS(NOW()) –
TO_DAYS(parution)) < 30
```

Exemple avancé : jointure & somme

produits

id	*classe*	*libelle*	*prix_ht*
01245	SPI	McCALLAN	25.6
01246	SPI	ARDBERG	30.8
01247	SPI	LAGAVULLIN	32.5

```
SELECT DISTINCT produits.libelle,
SUM(ventes.qte * produits.prix_ht) AS total
FROM produits, ventes
WHERE produits.id = ventes.produit
GROUP BY produits.libelle
ORDER BY total
```

ventes

id	*produit*	*qte*
00001256	01246	5
00001257	01246	2
00001258	01245	3
00001259	111111	30
00001260	01245	7
00001261	01247	1

libelle	*total*
LAGAVULIN	32.50
ARDBERG	215.60
McCALLAN	265.00

Fonctions arithmétiques

Fonction	Description
COUNT([DISTINCT]x,y,…)	Décompte des tuples du résultat par projection sur le ou les attributs spécifiés (ou tous avec '*'). L'option DISTINCT élimine les doublons.
MIN(x), MAX(x), AVG(x), SUM(x)	Calculent respectivement le minimum, le maximum, la moyenne et la somme des valeurs de l'attribut X.

Bruno B. SIMON

www.bbs-consultant.net

contact@bbs-consultant.com