

INTRODUCTION AU SQL

CONSOLE.LOG("BIENVENUE À TOUS");

LET'S BEGIN....

MCD

MODEL CONCEPTUEL DE DONNÉES

(PLAN DE CONSTRUCTION)

- MEMBRE
- PRODUIT
- COMMANDE

SGBD

SYSTEM DE GESTION DE BASE DE DONNÉES
(EXPLOITE LES DONNÉES)
MYSQL / MARIADB

SGBD

SERVEUR WEB : XAMPP

BDD

**BASE DE DONNÉES
(EMPLACEMENT DES DONNÉES SAUVEGARDÉES)**

SQL

STRUCTURED QUERY LANGUAGE
(COMMUNIQUE AVEC LE SGBD ET LES BDD)

CHACUN SON ROLE

HTML / CSS
JS / JQUERY
PHP
SQL

EXEMPLE

company_name

first_name

last_name

cars_id_car

address

postal_code

city

telephone

siret

type

Créer Client

RÉCUPÉRATION DES DONNÉES

PHP VA PERMETTRE DE RÉCUPÉRER LES SAISIES

ENREGISTRER LES DONNÉES

UNE REQUÊTE SQL VA PERMETTRE D'ENREGISTRER LES DONNÉES EN BASE DE DONNÉES

ENREGISTRER LES DONNÉES

Server: localhost:8889 » Database: shauto » Table: clients

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action			
	1	id_client	int(11)			No	None		AUTO_INCREMENT				
	2	first_name	varchar(70)	utf8_general_ci		No	None						
	3	last_name	varchar(70)	utf8_general_ci		No	None						
	4	address	varchar(100)	utf8_general_ci		No	None						
	5	postal_code	varchar(5)	utf8_general_ci		No	None						
	6	city	varchar(70)	utf8_general_ci		No	None						
	7	telephone	int(11)			No	None						
	8	type	enum('0', '1')	utf8_general_ci		No	None						
	9	siret	varchar(15)	utf8_general_ci		Yes	NULL						
	10	company_name	varchar(45)	utf8_general_ci		Yes	NULL						
	11	cars_id_car	int(11)			No	None						

Check all With selected: Browse Change Drop Primary Unique Index Fulltext

Print Propose table structure Move columns Normalize

ENREGISTRER LES DONNÉES

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT * FROM `clients`
```

Show all | Number of rows: 100 Filter rows: Search this table

+ Options

	id_client	first_name	last_name	address	postal_code	city	telephone	type	siret	company
<input type="checkbox"/>	1	Elazhar	Oussama	RUE DU MOULIN DE PIERRE	78270	BONNIERES SUR SEINE	783154495	1	80141165300026	

Show all | Number of rows: 100 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

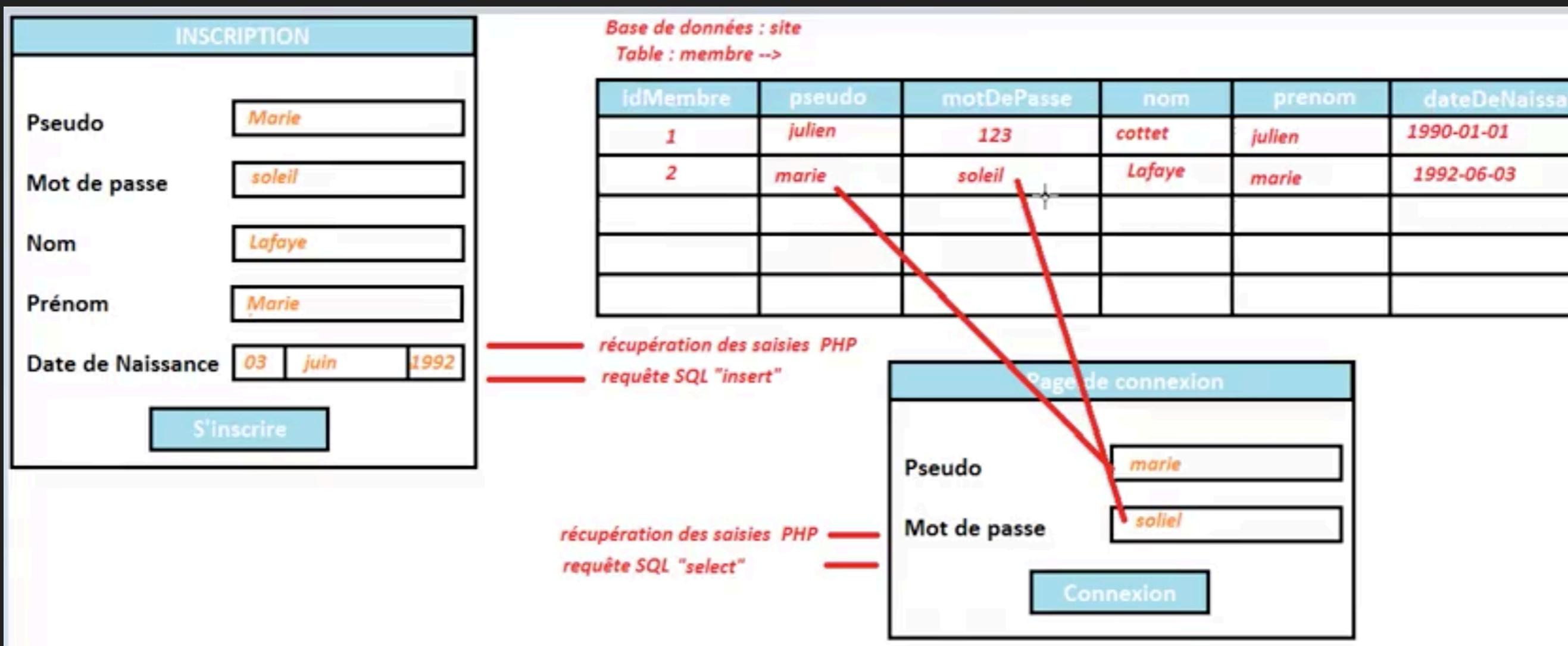
CHACUN SON RÔLE

LE HTML PERMET DE DÉCLARER LE FORMULAIRE
LE PHP PERMET DE RÉCUPÉRER LES DONNÉES
LE SQL PERMET D'INSÉRER LES DONNÉES EN BASE

LE SQL, UN CODE RÉUTILISABLE

SI LA REQUÊTE SQL ET LE FORMULAIRE HTML EST BIEN CODÉ VOUS ALLEZ POUVOIR RÉUTILISER LE CODE POUR ENREGISTRER D'AUTRES CLIENTS

AUTRE EXEMPLE



DES PAGES DYNAMIQUES

PARAMÈTRES ET URL(S)

DES PAGES DYNAMIQUES

Catalogue

catalogue.php

1 Tshirt Jaune  10 € Ajout au panier	2 Tshirt Bleu  15 € Ajout au panier	3 Tshirt Rouge  20 € Ajout au panier
4 Tshirt Gris  25 € Ajout au panier	5 Tshirt Noir  20 € Ajout au panier	6 Tshirt Blanc  15 € Ajout au panier

fiche_produit.php?id_produit=3

Fiche Produit

 **Tshirt Rouge**
Ref: TSHIRT_ROUGE

PROMO
NOUVEAUTE

[Lire la description complète](#)

Couleur :

Taille :

Quantité :

Livraison offerte !

En stock
-10 %

Ajouter au panier 

20 €

DES PAGES DYNAMIQUES

Catalogue

catalogue.php

1 Tshirt Jaune  10 € Ajout au panier	2 Tshirt Bleu  15 € Ajout au panier	3 Tshirt Rouge  20 € Ajout au panier
4 Tshirt Gris  25 € Ajout au panier	5 Tshirt Noir  30 € Ajout au panier	6 Tshirt Blanc  15 € Ajout au panier

fiche_produit.php?id_produit=1

Fiche Produit

 **Tshirt Rouge**
Ref: TSHIRT_ROUGE

PROMO
NOUVEAUTE

[Lire la description complète](#)

Couleur :

Taille :

Quantité :

En stock -10 %

Livraison offerte ! ↘

Ajouter au panier 

20 €



EXEMPLE EN LIGNE

← → C ⌂ zara.com/fr/fr/homme-chemises-l737.html?v1=1445099 ⭐ | 🔍 | 🎵 | 🚪 | 🚩 | ☰

Apps Enchères Etrangers Facturation Photo Assurance Sites vente Fournisseur Pièces immatriculation favoris du moment Compta »

RECHERCHER _____

SE CONNECTER AIDE 0

FEMME
HOMME

TOUT VOIR UNIES IMPRIMÉES + FILTRES

NOUVEAUTÉS

COLLECTION

MANTEAUX
BLOUSONS
BLAZERS
COSTUMES
PULLS | GILETS
PANTALONS
JEANS
CHEMISES

Tout Voir
Unies
Imprimées

T-SHIRTS
SWEATS
CHAUSSURES
SACS
ACCÉSOIRES



**RELAXED
FIT
SHIRT**

<https://www.zara.com/fr/fr/homme-chemises-l737.html?v1=1445099>

EXEMPLE EN LIGNE

zara.com/fr/fr/chemise-polo-effet-daim-comfort-p07545340.html?v1=38555106&v2=1445099

Apps Enchères Etrangers Facturation Photo Assurance Sites vente Fournisseur Pièces immatriculation favoris du moment Compta 0



RECHERCHER SE CONNECTER AIDE 0

CHEMISE POLO EFFET DAIM COMFORT

45,95 EUR

NOIR - 7545/340

Chemise coupe décontractée, col polo et manches courtes. Bas réglable sur les côtés. Fermeture avant

▼ Voir plus

M

L

XL Coming Soon

✉ M'avertir de la disponibilité

QUELLE EST MA TAILLE? 

CHAT

https://static.zara.net/photos///2020/V/0/2/p/7545/340/800/2/w/1120/7545340800_1_1.jpg?ts=1579537280796

URL RE WRITING

URLS

RÉCUPÉRATION DE L'URL EN PHP
EFFECTUER UNE REQUÊTE SQL AVEC LES PARAMÈTRES EN URL
REQUÊTE DE TYPE "SELECT"

IMAGES



IMAGES

REQUÊTE DE SÉLECTION

IL FAUDRA BOUCLER DANS LE RÉSULTAT DE LA REQUÊTE EN PHP POUR VARIABLISER LES VALEURS DE L'ATTRIBUT SRC POUR CHAQUE IMAGE.

CELA ÉVITERA DE GÉRER LES IMG UNE PAR UNE =>

<H1>\$TITRE<H1>

IMAGES

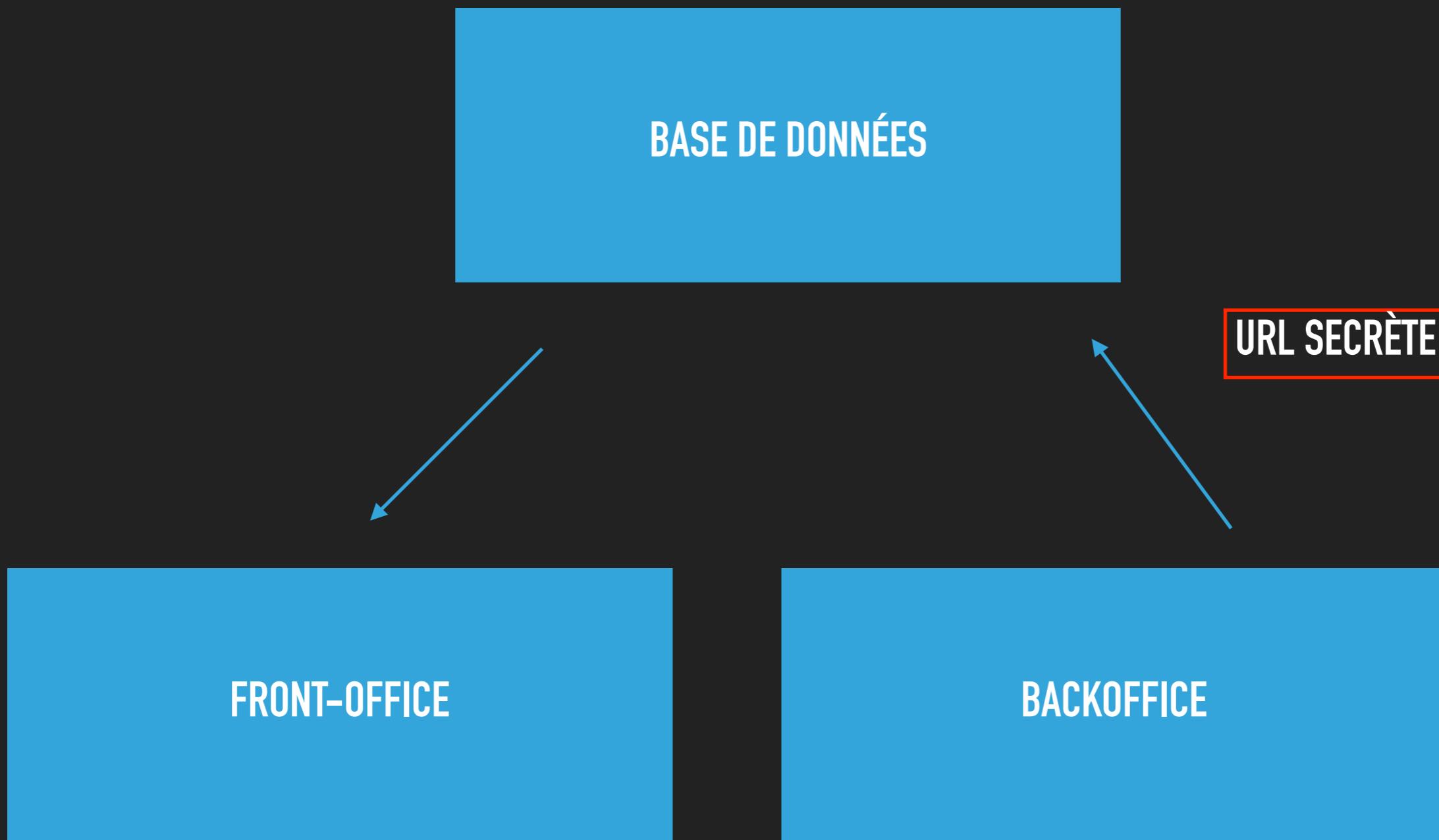
LE HTML NE CHANGE PAS
MAIS LE CONTENU DES BALISES SERA GÉRER AVEC PHP

IMAGES

DANS L'EXEMPLE IL Y A DEUX REQUÊTES:

UNE POUR RÉCUPÉRER UN VÊTEMENT
UNE POUR RÉCUPÉRER TOUS LES VÊTEMENTS

IMAGES



POURQUOI FAIRE UN BACKOFFICE?

POUR PERMETTRE PAR EXEMPLE À VOTRE CLIENT DE MODIFIER LE CONTENU ET OU IMAGES DE SON SITE DE MANIÈRE INDÉPENDANTE

COMMENT M'Y PRENDRE DU COUP?

LE CONTENU DE VOTRE SITE VA ÊTRE EN BASE DE DONNÉES
ET UNE VARIABLE VA VENIR REMPLIR L'ENSEMBLE DE VOS ÉLÈMENTS HTML

PLUS LES INFOS SONT EN BASE PLUS ON POURRA LES MODIFIER CÔTÉ BACKOFFICE

COMMENT M'Y PRENDRE DU COUP?

```
<BALISE> MON CONTENU </BALISE>
<BALISE> $MONCONTENU </BALISE>
```

COMMENT M'Y PRENDRE DU COUP?

SITE STATIQUE
VS
SITE DYNAMIQUE

FACEBOOK...LEPARISIEN...

LES DIFFÉRENTS TYPE DE REQUÊTES

SELECT (AFFICHAGE)

INSERT (AJOUT D'UN COMPTE UTILISATEUR)

UPDATE (MODIFICATION DU COMPTE UTILISATEUR)

DELETE (SUPPRESSION D'UN COMMENTAIRE)

LES TABLES
LES COLONES / CHAMPS
LES IDENTIFIANTS (CLÉ PRIMAIRE / PK PRIMARY KEY)
AUTO_INCREMENT
NULL/NOT NULL
LES RELATIONS
LES CARDINALICES
TABLE DE JOINTURE
CLÉ ÉTRANGÈRE
LES ENREGISTREMENTS
LES REQUÊTES

RAPPEL SUR LES TABLES D'ASSOCIATION

- id_conducteur
 - 1
 - 2
 - 3

= vehicule =

- id_vehicule
 - 780
 - 781
 - 782

= association_conducteur_vehicule

- id_association	- id_conducteur	- id_vehicule
36	3	781
37	2	780
38	1	782
39	1	780

LES ENREGISTREMENTS

LES DONNÉES À L'INTÉRIEUR DES TABLES

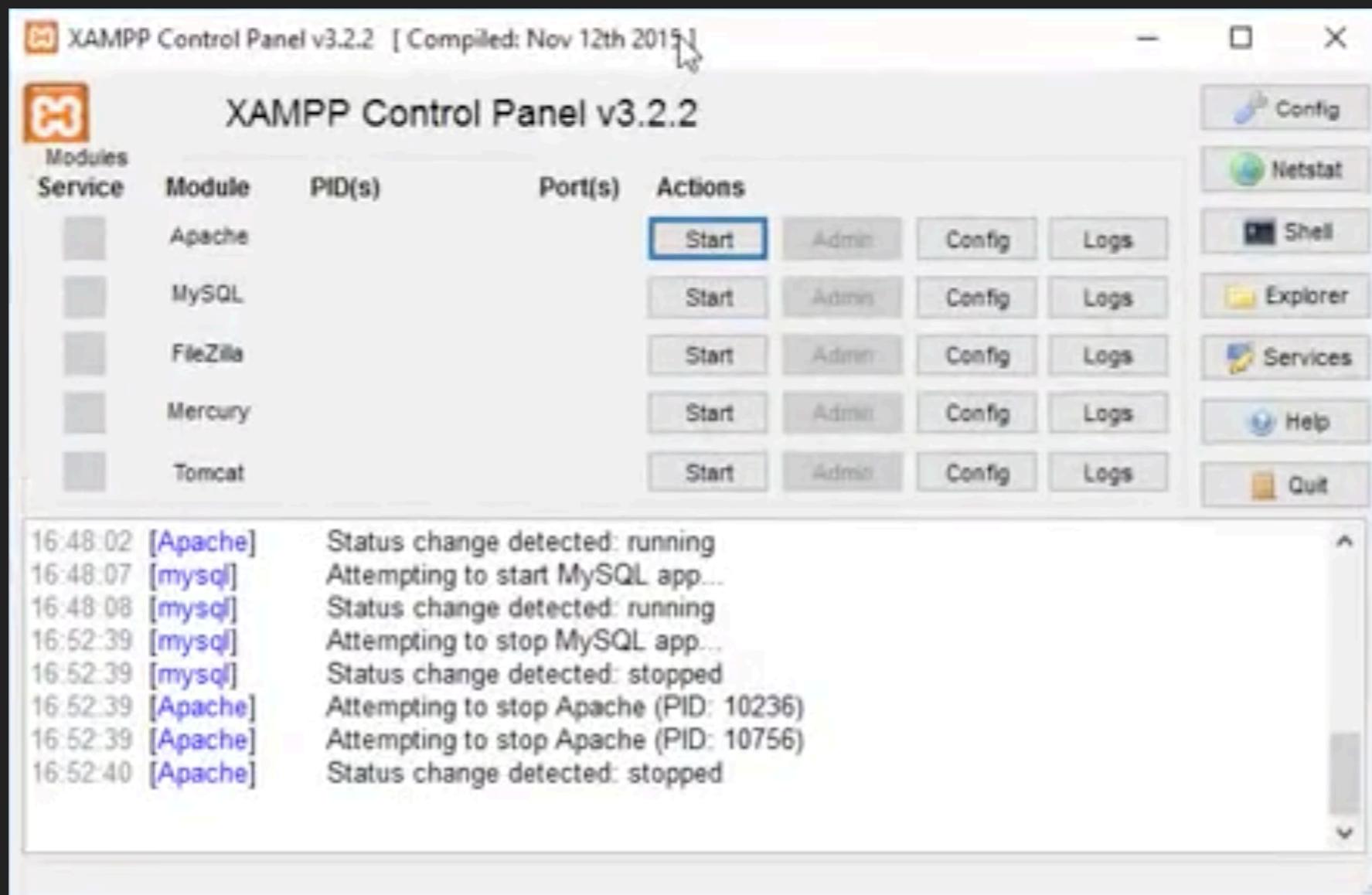
SERVER WEB

XAMPP

LES COMPOSANTS

SERVEUR APACHE
LE MYSQL
L'INTERPRETEUR PHP

SQL



SHELL



WS SH AUTO — bash — 80x24

```
Samihs-MBP:WS SH AUTO samiḥhabbani$ mysql.exe -u root --password
```

SHELL

```
Setting environment for using XAMPP for Windows.  
75@PC c:\xampp  
# mysql.exe -u root --password  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 2  
Server version: 10.1.21-MariaDB mariadb.org binary distribution  
  
Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]> SHOW DATABASES ;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| phpmyadmin |  
| test |  
+-----+  
5 rows in set (0.00 sec)  
  
MariaDB [(none)]> ■
```

MAC

BREW INSTALL MYSQL
BREW SERVICES START MYSQL
MYSQL -UROOT

MAJUSCULES ET MINUSCULES

LES MOTS CLEFS DU LANGAGE EN MAJUSCULES

LE RESTE EN MINUSCULE

(CONVENTIONS)

CRÉATION DE BASE DE DONNÉES

CREATE DATABASE ENTREPRISE

CRÉER DES TABLES

USE ENTREPRISE;
(UTILISATION DE LA BASE DE DONNÉES)

CRÉER DES TABLES

```
CREATE TABLE employes (
    id_employes INT NOT NULL AUTO_INCREMENT,
    prenom VARCHAR(20) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    sexe ENUM('m','f') NOT NULL,
    service VARCHAR(30) NOT NULL,
    date_embauche DATE NOT NULL,
    salaire INT NOT NULL,
    PRIMARY KEY (id_employes)
) ENGINE=InnoDB ;
```

MONTRER LES TABLES

SHOW TABLES;

MONTRER LES COLONNES

DESC EMPLOYES;

INSÉRER DES DONNÉES DANS UNE TABLE

```
INSERT INTO employes (id_employes, prenom, nom, sexe, service, date_embauche, salaire)
VALUES
(350, 'Jean-pierre', 'Laborde', 'm', 'direction', '2010-12-09', 5000),
(388, 'Clement', 'Gallet', 'm', 'commercial', '2010-12-15', 2300),
(415, 'Thomas', 'Winter', 'm', 'commercial', '2011-05-03', 3550),
(417, 'Chloe', 'Dubar', 'f', 'production', '2011-09-05', 1900),
(491, 'Elodie', 'Fellier', 'f', 'secretariat', '2011-11-22', 1600),
(509, 'Fabrice', 'Grand', 'm', 'comptabilite', '2011-12-30', 2900),
(547, 'Melanie', 'Collier', 'f', 'commercial', '2012-01-08', 3100),
(592, 'Laura', 'Blanchet', 'f', 'direction', '2012-05-09', 4500),
(627, 'Guillaume', 'Miller', 'm', 'commercial', '2012-07-02', 1900),
(655, 'Celine', 'Perrin', 'f', 'commercial', '2012-09-10', 2700),
(699, 'Julien', 'Cottet', 'm', 'secretariat', '2013-01-05', 1390),
(701, 'Mathieu', 'Vignal', 'm', 'informatique', '2013-04-03', 2500),
(739, 'Thierry', 'Desprez', 'm', 'secretariat', '2013-07-17', 1500),
(780, 'Amandine', 'Thoyer', 'f', 'communication', '2014-01-23', 2100),
(802, 'Damien', 'Durand', 'm', 'informatique', '2014-07-05', 2250),
(854, 'Daniel', 'Chevel', 'm', 'informatique', '2015-09-28', 3100),
(876, 'Nathalie', 'Martin', 'f', 'juridique', '2016-01-12', 3550),
(900, 'Benoit', 'Lagarde', 'm', 'production', '2016-06-03', 2550),
(933, 'Emilie', 'Sennard', 'f', 'commercial', '2017-01-11', 1800),
(990, 'Stephanie', 'Lafaye', 'f', 'assistant', '2017-03-01', 1775);
```

SELECTIONNER TOUS LES EMPLOYES

SELECT * FROM EMPLOYES;

SELECTIONNER DES COLONNES

```
SELECT PRENOM, NOM FROM EMPLOYES;
```

DISTINCT

SELECT DISTINCT(SERVICE) FROM EMPLOYES;

A CONDITION QUE

```
SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SERVICE = 'INFORMATIQUE';
```

BETWEEN

```
SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE DATE_EMBAUCHE BETWEEN '2015-01-01' AND  
'2019-01-27';
```

CURDATE()

```
SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE DATE_EMBAUCHE BETWEEN '2015-01-01' AND  
CURDATE();
```

LIKE

```
SELECT PRENOM FROM EMPLOYES WHERE PRENOM LIKE 'S%';
```

```
SELECT PRENOM FROM EMPLOYES WHERE PRENOM LIKE '%S';
```

```
SELECT PRENOM FROM EMPLOYES WHERE PRENOM LIKE '%S%';
```

EXEMPLE DE L'UTILISATION DE LIKE

```
-- logement
-- id_logement    titre      prix      superficie   cp
-- 1               appart1    800       27          75015
-- 2               appart2    1050      40          75017
-- 3               appart3    950       53          69003
-- 4               appart4    980       36          75010

-- Recherche :
-- prix max : 1000 €
-- superficie min : 30 m²
-- adresse : 75

SELECT * FROM logement WHERE prix <= 1000 AND superficie >= 30 AND cp LIKE '75%' ;
```

!=

SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SERVICE != 'INFORMATIQUE'

<> <= =>

SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SALAIRE > 3000;

SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SALAIRE >= 3000;

SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SALAIRE < 3000;

SELECT PRENOM, NOM, SERVICE FROM EMPLOYES WHERE SALAIRE <= 3000;

ORDER BY

SELECT PRENOM FROM EMPLOYES ORDER BY PRENOM ASC;

(ASCENDANT CROISSANT)

SELECT PRENOM FROM EMPLOYES ORDER BY PRENOM DESC;
(DESCENDANT DÉCROISSANT)

(PAR DÉFAUT ASC);

ORDER BY

Mes filtres

Smartphone

Tout supprimer

Trier par

- Prix croissant
- Prix décroissant
- Meilleure note
- Nouveauté
- Tri par défaut

3 877 modèles

VENDU PAR

DARTY 

Vendeurs partenaires

PRIX

4 € 23 999 €

CONFIRMER

MODE DE LIVRAISON

Retrait en magasin

Retrait en relais

Livraison à domicile

SMARTPHONE SAMSUNG GALAXY A10 32GO NOIR  4,7/5 159,00€*

SMARTPHONE SAMSUNG GALAXY S8 NOIR CARBONE  4,6/5 288,00€

SMARTPHONE SAMSUNG GALAXY A20E 32GO NOIR  4,8/5 189,00€*

Résultats par page 10 - 30 - 50 - 200 1 - 2 - 3 - 4 - 5 Page suivante

Smartphone DEALMARCHE HOUSSE FOLIO POUR MOBILE IPHONE X/XS HARD CASE  4,97€ Vendu par C Promo ★★★★☆ État : neuf

En stock 

ORDER BY

```
SELECT * FROM PRODUITS WHERE CATEGORIE = "TELEPHONE" AND PRIX <= 315 ORDER BY PRIX ASC;
```

TRI SECONDAIRE

```
SELECT * FROM EMPLOYES ORDER BY SALAIRE ASC, PRENOM DESC;
```

SELECT LIMITÉ

```
SELECT * FROM EMPLOYES ORDER BY SALAIRE ASC, PRENOM DESC LIMIT 0,5;
```

```
SELECT * FROM EMPLOYES ORDER BY SALAIRE ASC, PRENOM DESC LIMIT 5,5;
```

```
SELECT * FROM EMPLOYES ORDER BY SALAIRE ASC, PRENOM DESC LIMIT 10,5;
```

(ENREGISTREMENT DE DÉPART ET COMBIEN D'ENREGISTREMENTS NOUS VOULONS À PARTIR DE
L'ENREGISTREMENT DE DÉPART)

ONLINE

A red arrow points from the word "ONLINE" down to the screenshot of the darty.com website.

darty.com/nav/extra/list?p=200&s=prix_asc&cat=693&npk=1

Mes filtres

Smartphone Tout supprimer

VENDU PAR

DARTY  

Vendeurs partenaires

PRIX

4 € 23 999 €

Résultats par page 10 - 30 - 50 - **200** 1 - 2 - 3 - 4 - 5 Page suivante

CONFIRMER

MODE DE LIVRAISON

Retrait en magasin Retrait en relais

Trier par **Prix croissant** ? 3 877 modèles

IPHONE X IPHONE 8 GALAXY S9 SAMSUNG GALAXY SAMSUNG S7 HUAWEI P20 GOOGLE PIXEL 4

SMARTPHONE SAMSUNG GALAXY A10 32GO NOIR  4,7/5 159,00€*

SMARTPHONE SAMSUNG GALAXY S8 NOIR CARBONE  4,6/5 288,00€

SMARTPHONE SAMSUNG GALAXY A20E 32GO NOIR  4,8/5 189,00€*

Smartphone DEALMARCHE HOUSSE FOLIO POUR MOBILE IPHONE X/XS HARD CASE  4,97€ Vendu par C Promo ★★★★☆ État : neuf

OPÉRATIONS

```
SELECT PRENOM , (SALAIRE*12) FROM EMPLOYES;
```

AS POUR ALIAS

```
SELECT PRENOM , (SALAIRE*12) AS 'SALAIRE ANNUEL' FROM EMPLOYES;
```

SUM

```
SELECT SUM(SALAIRE*12) AS 'SALAIRE ANNUEL' FROM EMPLOYES;
```

AVG

```
SELECT AVG(SALAIRE) FROM EMPLOYES;
```

ROUND

```
SELECT ROUND(AVG(SALAIRE)) FROM EMPLOYES;
```

ROUND

```
SELECT ROUND(AVG(SALAIRE),2) FROM EMPLOYES;
```

COUNT

```
SELECT COUNT(PRENOM) FROM EMPLOYES;
```

MIN / MAX

SELECT MIN(SALAIRE) FROM EMPLOYES;

SELECT MAX(SALAIRE) FROM EMPLOYES;

ATTENTION !

SELECT PRENOM, MIN(SALAIRE) FROM EMPLOYES;

**(RÉSULTAT INCOHÉRENT, MIN PERMET D'ISOLER LA VALEUR LA PLUS FAIBLE
MAIS PAS DE RÉCUPÉRER LE PRÉNOM ASSOCIÉ À CETTE VALEUR)**

REQUÊTE IMBRIQUÉE

```
SELECT PRENOM FROM EMPLOYES WHERE SALAIRE = MIN(SALAIRE)
```

(CELA NE FONCTIONNE PAS !)

LA SOLUTION : REQUÊTE IMBRIQUÉE

```
SELECT PRENOM, NOM FROM EMPLOYES WHERE SALAIRE = (SELECT MIN(SALAIRE) FROM EMPLOYES);
```

=

```
SELECT PRENOM, NOM FROM EMPLOYES WHERE SALAIRE = 1390;
```

IN

```
SELECT PRENOM, NOM FROM EMPLOYES WHERE SERVICE IN("INFORMATIQUE","COMPTABILITE");
```

NOT IN

```
SELECT PRENOM, NOM FROM EMPLOYES WHERE SERVICE NOT IN("INFORMATIQUE", "COMPTABILITE");
```

GROUP BY

SELECT SERVICE, COUNT(*) FROM EMPLOYES;

VS

SELECT SERVICE, COUNT(*) FROM EMPLOYES GROUP BY SERVICE;

HAVING

```
SELECT SERVICE, COUNT(*) FROM EMPLOYES GROUP BY SERVICE HAVING COUNT(*) > 1;
```

```
SELECT SERVICE, COUNT(PRENOM) FROM EMPLOYES GROUP BY SERVICE HAVING COUNT(PRENOM) > 1;
```

INSERT

```
INSERT INTO EMPLOYES (ID_EMPLOYES, PRENOM, NOM, SEXE, SERVICE, DATE_EMBAUCHE, SALAIRE) VALUES  
(350, 'JEAN-PIERRE', 'LABORDE', 'M', 'DIRECTION', '2010-12-09', 5000).
```

ATTENTION AUX DUPLICATIONS DE PK

LA COMMANDE PRÉCÉDENTE NE POURRA PAS ÊTRE EXÉCUTÉE DEUX FOIS CAR LA PK ID_EMPLOYES 350 EXISTE DÉJÀ.

PLUSIEURS FAÇONS D'INSÉRER EN BASE

```
INSERT INTO EMPLOYES VALUES(  
    (NULL, 'JEAN-PIERRE', 'LABORDE', 'M', 'DIRECTION', '2010-12-09', 5000),
```

```
INSERT INTO EMPLOYES (ID_EMPLOYES, PRENOM, NOM, SEXE, SERVICE, DATE_EMBAUCHE, SALAIRE) VALUES  
    (350, 'JEAN-PIERRE', 'LABORDE', 'M', 'DIRECTION', '2010-12-09', 5000),
```

```
INSERT INTO EMPLOYES VALUES(  
    (350, 'JEAN-PIERRE', 'LABORDE', 'M', 'DIRECTION', '2010-12-09', 5000),
```

UPDATE

```
UPDATE EMPLOYES SET SALAIRE = 1291 WHERE ID_EMPLOYES = 699;
```

```
UPDATE EMPLOYES SET SALAIRE = 1291, SERVICE = 'INFORMATIQUE' WHERE ID_EMPLOYES = 699;
```

```
UPDATE EMPLOYES SET SALAIRE = (SALAIRE +100) WHERE SERVICE = 'COMMERCIAL';
```

DELETE

`DELETE FROM EMPLOYES WHERE ID_EMPLOYES = 699;`

`DELETE FROM EMPLOYES WHERE ID_EMPLOYES IN(992, 993);`

`DELETE FROM EMPLOYES WHERE ID_EMPLOYES > 991;`

`DELETE FROM EMPLOYES WHERE SERVICE = 'INFORMATIQUE' AND ID_EMPLOYES != 802`

SQL

EXO PRATIQUE

SQL

ÉXO PRATIQUE

SQL

```
CREATE TABLE abonne (
    id_abonne INT(3) NOT NULL AUTO_INCREMENT,
    prenom VARCHAR(15) NOT NULL,
    PRIMARY KEY (id_abonne)
) ENGINE=InnoDB ;

INSERT INTO abonne (id_abonne, prenom) VALUES
(1, 'Guillaume'),
(2, 'Benoit'),
(3, 'Chloe'),
(4, 'Laura');

CREATE TABLE livre (
    id_livre INT(3) NOT NULL AUTO_INCREMENT,
    auteur VARCHAR(25) NOT NULL,
    titre VARCHAR(30) NOT NULL,
    PRIMARY KEY (id_livre)
) ENGINE=InnoDB ;
```

SQL

```
CREATE TABLE livre (
    id_livre INT(3) NOT NULL AUTO_INCREMENT,
    auteur VARCHAR(25) NOT NULL,
    titre VARCHAR(30) NOT NULL,
    PRIMARY KEY (id_livre)
) ENGINE=InnoDB ;

INSERT INTO livre (id_livre, auteur, titre) VALUES
(100, 'GUY DE MAUPASSANT', 'Une vie'),
(101, 'GUY DE MAUPASSANT', 'Bel-Ami '),
(102, 'HONORE DE BALZAC', 'Le pre Goriot'),
(103, 'ALPHONSE DAUDET', 'Le Petit chose'),
(104, 'ALEXANDRE DUMAS', 'La Reine Margot'),
(105, 'ALEXANDRE DUMAS', 'Les Trois Mousquetaires');

CREATE TABLE emprunt (
    id_emprunt INT(3) NOT NULL AUTO_INCREMENT,
    id_livre INT(3) DEFAULT NULL,
    id_abonne INT(3) DEFAULT NULL,
    date_sortie DATE NOT NULL,
    date_rendu DATE DEFAULT NULL,
    PRIMARY KEY (id_emprunt)
) ENGINE=InnoDB ;

INSERT INTO emprunt (id_emprunt, id_livre, id_abonne, date_sortie, date_rendu) VALUES
(1, 100, 1, '2016-12-07', '2016-12-11'),
(2, 101, 2, '2016-12-07', '2016-12-18'),
(3, 100, 3, '2016-12-11', '2016-12-19'),
(4, 103, 4, '2016-12-12', '2016-12-22'),
(5, 104, 1, '2016-12-15', '2016-12-30'),
(6, 105, 2, '2017-01-02', '2017-01-15'),
(7, 105, 3, '2017-02-15', NULL),
(8, 100, 2, '2017-02-20', NULL);
```

SQL

```
CREATE TABLE emprunt (
    id_emprunt INT(3) NOT NULL AUTO_INCREMENT,
    id_livre INT(3) DEFAULT NULL,
    id_abonne INT(3) DEFAULT NULL,
    date_sortie DATE NOT NULL,
    date_rendu DATE DEFAULT NULL,
    FOREIGN KEY (id_livre) REFERENCES livre(id_livre),
    FOREIGN KEY (id_abonne) REFERENCES abonne(id_abonne),
    PRIMARY KEY  (id_emprunt)
) ENGINE=InnoDB ;
```

SQL

ATTENTION SI VOUS RAJOUTER DES CLÉS ÉTRANGÈRES DANS LA TABLE EMPRUNT, IL FAUDRA CRÉER LES LIVRES ET LES ABONNÉS AVANT CAR SINON VOUS ALLEZ RÉFÉRENCE À DES LIVRES ET DES ABONNÉS QUI N'EXISTE PAS.

LES FOREIGN KEY EN SQL C'EST UNE SÉCURITÉ SUPPLÉMENTAIRE POUR ÉVITER DES DONNÉES ERRONÉES.

SANS CLÉ ÉTRANGÈRES NOUS AURIONS PU NOUS RETROUVER AVEC DES EMPRUNTS FAITS SUR DES LIVRES ET DES ABONNÉS INEXISTANTS. (C'EST UNE ERREUR SYSTEM BONNE, ET VOULU).

SQL

POURQUOI UNE FOREIGN KEY?

SQL

PEUT-ON CRÉER LES TABLES SANS FK?

SQL

REQUÊTES IMBRIQUÉES

C'EST L'HEURE DE REQUÉTER

NOUS POUVONS REPRENDRE SUR L'EXO SUR LA
BIBLIOTHÈQUE

ATTENTION

```
SELECT ID_LIVRE FROM EMPRUNT DATE_RENDER = NULL;
```

(NE PRODUIT PAS D'ERREUR MAIS CE N'EST PAS COMME CELA QUE NOUS MANIPULONS LA VALEUR NULL EN SQL)

IS NULL

POUR TROUVER UNE VALEUR NULL NOUS UTILISONS LE “IS NULL”

```
SELECT ID_LIVRE FROM EMPRUNT DATE_RENDER IS NULL;
```

REQUÊTE IMBRIQUÉE

JE SOUHAITE RÉCUPÉRER LE TITRE DES LIVRES QUI N'ONT PAS ENCORE ÉTÉ RENDU:

```
SELECT TITRE FROM LIVRE WHERE ID_LIVRE IN (SELECT ID_LIVRE FROM EMPRUNT WHERE DATE_RENDER IS  
NULL );
```

ICI ON ISOLE D'ABORD L'ID DES LIVRES DANS LA TABLE EMPRUNT QUI N'ONT PAS ENCORE ÉTÉ RENDU.

TABLE DE JOINTURE

```
SELECT PRENOM FROM ABONNE WHERE ID_ABONNE IN (SELECT ID_ABONNE FROM EMPRUNT WHERE  
DATE_RENDERU IS NULL);
```

SQL

AFFICHER LES NUMÉROS DE LIVRE QUE CHLOÉ À EMPRUNTER :

À L'OEIL NU :

```
SELECT ID_LIVRE FROM EMPRUNT WHERE ID_ABONNE = 3;
```

AVEC UNE REQUÊTE IMBRIQUÉE:

```
SELECT ID_LIVRE FROM EMPRUNT WHERE ID_ABONNE IN (SELECT ID_ABONNE FROM ABONNE WHERE PRENOM  
= 'CHLOE');
```

SQL

AFFICHER LES PRÉNOMS DES ABONNÉS AYANT EMPRUNTÉ UN LIVRE À LA DATE DU 07/12/2016

```
SELECT PRENOM FROM ABONNE WHERE ID_ABONNE IN (SELECT ID_ABONNE FROM EMPRUNT WHERE  
DATE_SORTIE = '07/12/2016');
```

SQL

AFFICHER LE NOMBRE DE LIVRE EMPRUNTÉ PAR GUILLAUME

SQL

```
SELECT COUNT(ID_LIVRE) FROM EMPRUNT WHERE ID_ABONNE IN(SELECT ID_ABONNE FROM ABONNE WHERE PRENOM = 'GUILLAUME');
```

SQL

ET AVEC UN ALIAS?

```
SELECT COUNT(ID_LIVRE) AS 'NBRE DE LIVRE EMPRUNTÉ PAR GUILLAUME' FROM EMPRUNT WHERE  
ID_ABONNE IN(SELECT ID_ABONNE FROM ABONNE WHERE PRENOM = 'GUILLAUME');
```

SQL

ET SI ON DEVAIT PASSER PAR LES 3 TABLES?

AFFICHER LE TITRE DES LIVRES QUE CHLOÉ À EMPRUNTER !

COMMENT S'Y PRENDRE?

AFFICHER L'ID_ABONNE DE CHLOE
AFFICHER LES IDS DES LIVRES EMPRUNTÉS PAR L'ID ABONNÉ DE CHLOÉ
AFFICHER LES TITRES DE CES ID LIVRES TROUVÉS.

RESPONSE

```
SELECT TITRE FROM LIVRE WHERE ID_LIVRE IN (
```

```
    SELECT ID_LIVRE FROM EMPRUNT WHERE ID_ABONNE IN (
```

```
        SELECT ID_ABONNE FROM ABONNE WHERE PRENOM = 'CHLOE');
```

LA LISTE DES LIVRES QU'ELLE N'A PAS ENCORE EMPRUNTÉ

```
SELECT TITRE FROM LIVRE WHERE ID_LIVRE NOT IN (
```

```
    SELECT ID_LIVRE FROM EMPRUNT WHERE ID_ABONNE IN (
```

```
        SELECT ID_ABONNE FROM ABONNE WHERE PRENOM = 'CHLOE' ));
```

EXO SUPPLÉMENTAIRE

CONNAITRE LE NOM DES ABONNÉS QUI ONT EMPRUNTÉ LE LIVRE D'ALPHONSE DAUDET.

SQL

JOINTURE

SQL

REQUÊTES IMBRIQUÉES VS JOINTURE

SQL

JOINTURE POUR RELATIONS ENTRE LES TABLES

LE RÉSULTAT EST DANS LE RÉSULTAT FINALE

L'IMBRIQUÉE N'EST PAS POSSIBLE DANS TOUS LES CAS.
ELLE N'EST POSSIBLE QUE QUAND ON VEUT LES COLONNES DE LA MÊME TABLE.

EX : TITRE DU LIVRE, DATE SORTIE LIVRE ET LE PRÉNOM DE L'ABONNÉ QUI L'A EMPRUNTÉ
-> RÉSULTATS ICI DE TROIS TABLES DIFFÉRENTES DONC SEULEMENT AVEC JOINTURE.

SQL

POURQUOI PAS TOUT LE TEMPS DES JOINTURES?

REQUÊTE IMBRIQUÉE SONT PLUS RAPIDE ET MOINS LOURDES. (SITE FORT TRAFIC)

POSSIBLE QUE EN ENTREPRISE LES REQUÊTES SOIT FAITE EN IMBRIQUÉE.

REQUÊTE DE JOINTURE

AFFICHER LES DATES DE SORTIES ET D'ENTRÉE DE GUILLAUME POUR RENDRE UN LIVRE.

SQL

```
SELECT A.PRENOM, E.DATE_SORTIE, E.DATE_RENDER FROM ABONNE A, EMPRUNT E WHERE A.ID_ABONNE =  
E.ID_ABONNE AND A.PRENOM = 'GUILLAUME';
```

SQL

A ET E SONT DES PRÉFIXES

DÉCOMPOSITION

LA PREMIÈRE LIGNE (SELECT) EST LES INFOS QUE L'ON SOUHAITE AFFICHER
LA DEUXIÈME (FROM) CORRESPOND AUX TABLES DONT J'AI BESOIN
LA TROISIÈME LIGNE (WHERE) CORRESPOND À LA CONDITION

AMBIGUITÉ ET PRÉFIXES

```
SELECT A.PRENOM, E.DATE_SORTIE, E.DATE_RENDERU FROM ABONNE A, EMPRUNT E WHERE A.ID_ABONNE =  
E.ID_ABONNE AND A.PRENOM = 'GUILLAUME';
```

EXO COMPLÉMENTAIRE

AFFICHER LES MOUVEMENTS DES LIVRES ALPHONSE DAUDET (QUAND EST-CE QU'IL ONT ÉTÉ EMPRUNTÉS ET QUAND EST-CE QU'ILS ONT ÉTÉ RENDUS)

EXO COMPLÉMENTAIRE

AFFICHER LE OU LES ABONNÉS QUI ONT EMPRUNTÉ LE LIVRE “UNE VIE” SUR L’ANNÉE 2016.

EXO COMPLÉMENTAIRE

AFFICHER LE NOMBRE DE LIVRE EMPRUNTÉ PAR CHAQUE ABONNÉ

EXO COMPLÉMENTAIRE

AFFICHER LE NOMBRE DE LIVRE À RENDRE POUR CHAQUE ABONNÉ

EXO COMPLÉMENTAIRE

QUI A EMPRUNTÉ QUOI ET QUAND?

POUR INFO

LE SENS DES PRÉFIXES N'IMPORTE PAS DANS LES CONDITIONS

CORRECTION

```
-- Afficher les mouvements des livres d'Alphonse Daudet (quand est-ce  
qu'ils ont été empruntés et quand est-ce qu'ils ont été rendus)  
SELECT e.date_sortie, e.date_rendu  
FROM emprunt e, livre l  
WHERE e.id_livre = l.id_livre  
AND l.auteur = 'Alphonse Daudet' ;I
```

CORRECTION

```
-- Afficher l'abonné (les abonnés) ayant emprunté le livre "Une Vie"  
sur l'année 2016  
SELECT a.prenom, e.date_sortie  
FROM abonne a, emprunt e, livre l  
WHERE a.id_abonne = e.id_abonne  
AND l.id_livre = e.id_livre  
AND l.titre = 'Une Vie'  
AND e.date_sortie BETWEEN '2016-01-01' AND '2016-12-31' ;
```

SQL

```
-- Afficher le nombre de livre emprunté par chaque abonné
SELECT a.prenom, COUNT(e.id_livre)
FROM abonne a, emprunt e
WHERE a.id_abonne = e.id_abonne
GROUP BY a.prenom ;
```

SQL

```
-- Afficher le nombre de livre a rendre pour chaque abonné
SELECT a.prenom, COUNT(e.id_livre)
FROM abonne a, emprunt e
WHERE a.id_abonne = e.id_abonne
AND date_rendu IS NULL
GROUP BY a.prenom ;
```

SQL

```
-- Qui a emprunté Quoi et Quand
-- résultat : (qui)prenom - (quoi)titre - (quand)date_sortie
SELECT a.prenom, l.titre, e.date_sortie
FROM abonne a, livre l, emprunt e
WHERE a.id_abonne = e.id_abonne
AND e.id_livre = l.id_livre ;    I
```

SQL

```
-- Qui a emprunté quoi ? prenom + id_livre

SELECT a.prenom, e.id_livre
FROM abonne a, emprunt e
WHERE a.id_abonne = e.id_abonne ;
```

LEFT JOIN

ON VA CRÉER UN NOUVEL ABONNÉ QUI N'A PAS FAIT D'EMPRUNT ET ON VA VOIR CE QU'IL SE PASSE AU NIVEAU DES RÉSULTATS SI ON FAIT UN LEFT JOIN.

SQL

```
INSERT INTO abonne (prenom) VALUES ('alexandre') ;  
  
SELECT * FROM abonne ;  
  
SELECT a.prenom, e.id_livre I  
FROM abonne a LEFT JOIN emprunt e  
ON a.id_abonne = e.id_abonne ;
```

SQL

```
SELECT a.prenom, e.id_livre  
FROM abonne a LEFT JOIN emprunt e  
ON a.id_abonne = e.id_abonne ;  
I
```

SQL

LE WHERE DEVIENT ON

TRADUCTION

RAPATRIE MOI TOUTES LES DONNÉES DE LA TABLE ABONNE MÊME SI IL N'Y A PAS DE CORRESPONDANCE. C'EST À DIRE MÊME SI ILS N'ONT PAS FAIT D'EMPRUNT.

SQL

```
+-----+-----+
| prenom | id_livre |
+-----+-----+
| Guillaume | 108 |
| Guillaume | 104 |
| Benoit | 101 |
| Benoit | 105 |
| Benoit | 108 |
| Chloe | 108 |
| Chloe | 105 |
| Laura | 103 |
| alexandre | NULL |
+-----+-----+
9 rows in set (0.00 sec)

MariaDB [bibliotheque]>
```

SQL

ICI IL A BIEN RÉCUPÉRER TOUS LES ABONNÉS MÊME SI ILS N'ONT RIEN EMPRUNTÉ
C'EST À DIRE MÊME SI ILS N'ONT PAS DE PRÉSENCE DANS LA TABLE EMPRUNT.

AUTRES EXEMPLES AVEC LE LEFT JOIN

---	membre			
---	id_membre	pseudo	adresse	
	-- 1	julien	paris	
	-- 2	amandine	marseille	
---	produit			
---	id_produit	prix	titre	
	-- 880	50	pull	
	-- 885	70	chemise	
	-- 996	90	costume	
---	commande			
---	id_commande	id_membre	id_produit	date
	-- 1030	NULL	885	2015
	-- 1689	NULL	996	2015
	-- 1853	2	996	2015

SQL

DANS L'EXEMPLE PRÉCÉDENT, SI UN MEMBRE EST SUPPRIMÉ, SA CLÉ ÉTRANGÈRE DANS LA TABLE COMMANDE DEVRA ÉGALEMENT ÊTRE SUPPRIMÉ. DANS LE CADRE D'UN LEFT JOIN, ON AFFICHERA QUAND MÊME LES COMMANDES QUI AURONT UN ID_Membre NULL.

LEFT JOIN

```
SELECT c.* , m.*  
FROM commande c , membre m  
WHERE c.id_membre = m.id_membre ;
```

```
SELECT c.* , m.*  
FROM commande c LEFT JOIN membre m  
ON c.id_membre = m.id_membre ;
```

UNION

POUR FUSIONNER LES DONNÉES EN UNE SEULE

SQL

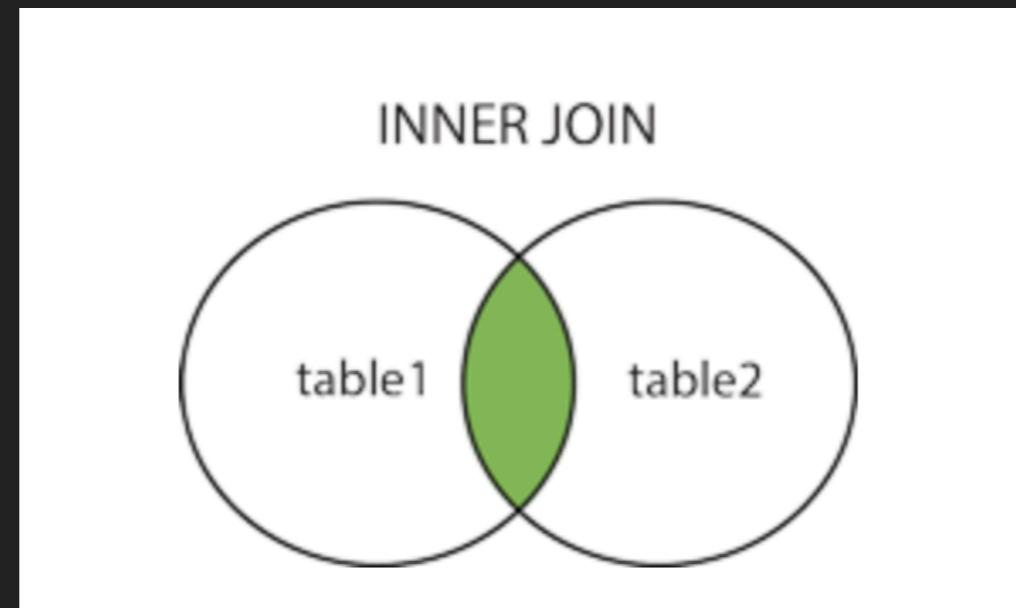
```
SELECT auteur FROM livre
UNION
SELECT prenom FROM abonne ;
-- UNION ALL
-- UNION DISTINCT
```

UNION ALL NOUS VERRONS LES DOUBLONS (SI DEUX PRÉNOM IDENTIQUES)
UNION DISTINCT NOUS VERRONS QU'UN SEUL PRÉNOM SI IL EST EN DOUBLON.

INNER JOIN

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

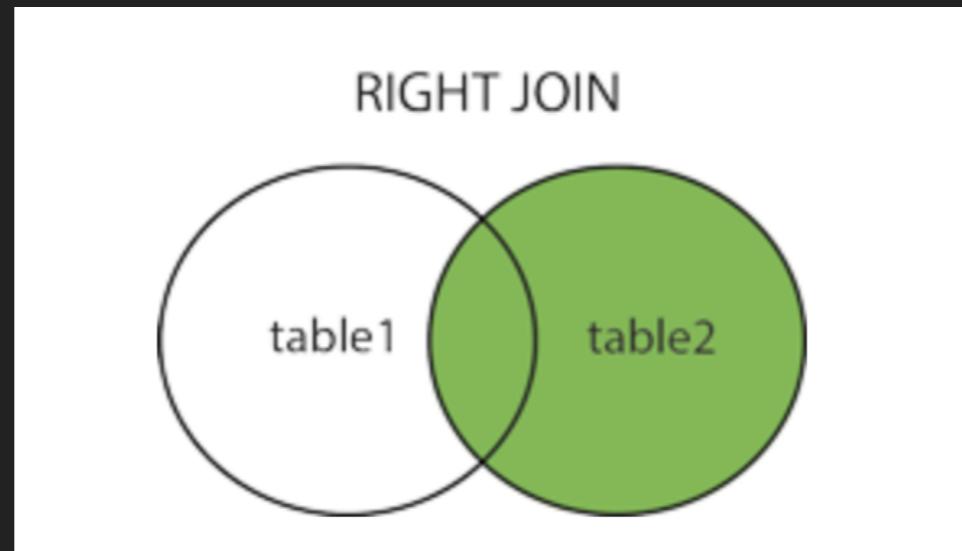
INNER JOIN



RIGHT JOIN

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

RIGHT JOIN



SQL

FUNCTIONS

FUNCTIONS

SELECT CURDATE();
-- DATE COURANTE

SELECT CURTIME();
-- HEURE COURANTE

SELECT NOW();
-- DATE + HEURE COURANTE

FUNCTIONS

```
SELECT ID_EMPRUN, ID_LIVRE, ID_ABONNE, DATE_FORMAT(DATE_SORTIE, '%D/%M/%Y') FROM EMPRUNT;
```

```
SELECT ID_EMPRUN, ID_LIVRE, ID_ABONNE, DATE_FORMAT(DATE_SORTIE, '%D/%M/%Y %H:%I:%S') FROM  
EMPRUNT;
```

```
SELECT ID_EMPRUN, ID_LIVRE, ID_ABONNE, DATE_FORMAT(DATE_SORTIE, '%D/%M/%Y %H:%I:%S') AS  
DATE_SORTIE FROM EMPRUNT;
```

FUNCTIONS

```
SELECT PASSWORD('BONJOUR');  
-- CRYPTAGE
```

FUNCTIONS

```
SELECT CONCAT(ID_ABONNE, ' ', PRENOM ) FROM ABONNE;
```

FUNCTIONS

MySQL Server MySQL Enterprise Workbench InnoDB Cluster MySQL NDB Cluster Connectors More

MySQL 8.0 Reference Manual / Functions and Operators version 8.0 ▾

Search this Manual 

Documentation Home

MySQL 8.0 Reference Manual

- Preface and Legal Notices
- General Information
- Installing and Upgrading MySQL
- Tutorial
- MySQL Programs
- MySQL Server Administration
- Security
- Backup and Recovery
- Optimization
- Language Structure
- Character Sets, Collations, Unicode
- Data Types
- **Functions and Operators**

Table of Contents

- 12.1 Function and Operator Reference
- 12.2 Type Conversion in Expression Evaluation
- 12.3 Operators
- 12.4 Control Flow Functions
- 12.5 Numeric Functions and Operators
- 12.6 Date and Time Functions
- 12.7 String Functions and Operators
- 12.8 What Calendar Is Used By MySQL?
- 12.9 Full-Text Search Functions
- 12.10 Cast Functions and Operators
- 12.11 XML Functions
- 12.12 Bit Functions and Operators
- 12.13 Encryption and Compression Functions
- 12.14 Locking Functions



FONCTION UTILISATEUR

MORCEAU DE CODE ISOLÉ À UN ENDROIT QUE L'ON PEUT RÉUTILISER

FONCTION UTILISATEUR

DELIMITER \$

LES REQUÊTES SQL NE DEVONT PLUS SE TERMINER PAR UN ; MAIS PAS UN \$.

FONCTION UTILISATEUR

```
-- FONCTION UTILISATEUR

DELIMITER $
CREATE FUNCTION salaire_brut_en_net(sal INT) RETURNS INT
COMMENT 'fonction permettant le conversion d\'un salaire'
NO SQL
BEGIN
    RETURN (sal*0.79);
END $
```

FONCTION UTILISATEUR

```
DELIMITER $  
CREATE FUNCTION salaire_brut_en_net(sal INT) RETURNS INT  
COMMENT 'fonction permettant le conversion d\'un salaire'  
NO SQL  
BEGIN  
..... ;  
..... ;  
..... ;  
..... ;  
..... ;  
RETURN (sal*0.79) ;  
END $
```

FONCTION UTILISATEUR

LE \$ PERMET DE DIRE QUE LA FONCTION S'ARRÈTERA NON PAS AU PREMIER POINT VIRGULE MAIS AU DERNIER DOLLARS.

PAR NATURE SQL EST LANCÉ DÉS QU'ELLE INTERPRÈTE UN POINT VIRGULE ET DONC LE DOLLARS PERMET DE CHANGER CE COMPORTEMENT DE SQL.

FONCTION UTILISATEUR

```
DELIMITER $  
CREATE FUNCTION salaire_brut_en_net(sal INT) RETURNS INT  
COMMENT 'fonction permettant le conversion d\'un salaire'  
NO SQL  
BEGIN  
    RETURN (sal*0.79);  
END $  
  
SELECT salaire_brut_en_net(2000) $
```

FONCTION UTILISATEUR

```
MariaDB [entreprise]> DELIMITER $  
MariaDB [entreprise]> CREATE FUNCTION salaire_brut_en_net(sal INT) RETURNS INT  
-> COMMENT 'fonction permettant la conversion d\'un salaire'  
-> NO SQL  
-> BEGIN  
-> RETURN (sal*0.79) ;  
-> END $  
Query OK, 0 rows affected (0.00 sec)  
  
MariaDB [entreprise]> SELECT salaire_brut_en_net(2000) $  
+-----+  
| salaire_brut_en_net(2000) |  
+-----+  
| 1580 |  
+-----+  
1 row in set (0.00 sec)  
  
MariaDB [entreprise]>
```

FONCTION UTILISATEUR

```
DELIMITER $  
CREATE FUNCTION salaire_brut_en_net(sal INT) RETURNS INT  
COMMENT 'fonction permettant la conversion d\'un salaire'  
NO SQL  
BEGIN  
    RETURN (sal*0.79);  
END $  
  
SELECT salaire_brut_en_net(2000) $
```

FONCTION UTILISATEUR

```
-- FONCTION UTILISATEUR

DELIMITER $
CREATE FUNCTION salaire_brut_en_net(s INT) RETURNS INT
COMMENT 'fonction permettant le conversion d\'un salaire'
NO SQL
BEGIN
    RETURN (s*0.79) ;
END $

SELECT salaire_brut_en_net(2000) $

INSERT INTO employes (prenom, salaire) VALUES ('robert',
salaire_brut_en_net(5000)) $
```

FONCTION UTILISATEUR

```
salaire_brut_en_net(5000); at line 1
MariaDB [entreprise]> INSERT INTO employes (prenom, salaire) VALUES ('robert', salaire_brut_en_net(5000)) $
Query OK, 1 row affected, 3 warnings (0.01 sec)

MariaDB [entreprise]> SELECT * FROM employes ;
-> $
+-----+-----+-----+-----+-----+-----+-----+
| id_employes | prenom | nom | sexe | service | date_embauche | salaire |
+-----+-----+-----+-----+-----+-----+-----+
| 350 | Jean-pierre | Laborde | m | direction | 2010-12-09 | 5000 |
| 388 | Clement | Gallet | m | commercial | 2010-12-15 | 2400 |
| 415 | Thomas | Winter | m | commercial | 2011-05-03 | 3650 |
| 417 | Chloe | Dubar | f | production | 2011-09-05 | 1900 |
| 491 | Elodie | Fellier | f | secretariat | 2011-11-22 | 1600 |
| 509 | Fabrice | Grand | m | comptabilite | 2011-12-30 | 2900 |
| 547 | Melanie | Collier | f | commercial | 2012-01-08 | 3200 |
| 592 | Laura | Blanchet | f | direction | 2012-05-09 | 4500 |
| 627 | Guillaume | Miller | m | commercial | 2012-07-02 | 2000 |
| 655 | Celine | Perrin | f | commercial | 2012-09-10 | 2800 |
| 739 | Thierry | Desprez | m | secretariat | 2013-07-17 | 1500 |
| 780 | Amandine | Thoyer | f | communication | 2014-01-23 | 2100 |
| 802 | Damien | Durand | m | informatique | 2014-07-05 | 2250 |
| 876 | Nathalie | Martin | f | juridique | 2016-01-12 | 3550 |
| 900 | Benoit | Lagarde | m | production | 2016-06-03 | 2550 |
| 933 | Emilie | Sennard | f | commercial | 2017-01-11 | 1900 |
| 990 | Stephanie | Lafaye | f | assistant | 2017-03-01 | 1775 |
| 994 | robert | | m | | 0000-00-00 | 3950 |
+-----+-----+-----+-----+-----+-----+
```

FONCTION UTILISATEUR

```
DELIMITER $  
CREATE FUNCTION salaire_brut_en_net(s INT) RETURNS INT  
COMMENT 'fonction permettant le conversion d\'un salaire'  
NO SQL  
BEGIN  
    RETURN (s*0.79) ;  
END $  
  
SELECT salaire_brut_en_net(2000) $  
  
INSERT INTO employes (prenom, salaire) VALUES ('robert',  
salaire_brut_en_net(5000)) $  
  
SELECT prenom, salaire_brut_en_net(salaire) FROM employes ;
```

FONCTION UTILISATEUR

' NO SQL ' PRÉCISE QUE NOUS N'AVONS PAS D'INSTRUCTIONS SQL NI DE REQUÊTE DANS LE CODE.

SQL

TABLES VIRTUELLES ET TEMPORAIRES

SQL

```
-- Table Virtuelle

USE bibliotheque ;

SELECT a.prenom, e.date_sortie, l.titre
FROM abonne a, emprunt e, livre l
WHERE a.id_abonne = e.id_abonne
AND e.id_livre = l.id_livre ;
```

SQL

```
-- Table Virtuelle (VIEW)

USE bibliotheque ;

CREATE VIEW vue_emprunt AS
    SELECT a.prenom, e.date_sortie, l.titre
    FROM abonne a, emprunt e, livre l
    WHERE a.id_abonne = e.id_abonne
    AND e.id_livre = l.id_livre ;
```

SQL

```
-- Table Virtuelle (VIEW)

USE bibliotheque ;

CREATE VIEW vue_emprunt AS
    SELECT a.prenom, e.date_sortie, l.titre
    FROM abonne a, emprunt e, livre l
    WHERE a.id_abonne = e.id_abonne
    AND e.id_livre = l.id_livre ;

SELECT * FROM vue_emprunt ;
```

I

SQL

UNE TABLE VIRTUELLE EST UNE TABLE QUI SE CRÉE À PARTIR D'UNE AUTRE TABLE ET QUI EST PRATIQUE POUR ISOLER DES RÉSULTATS.

ELLE POSSÈDE LES MÊMES DONNÉES QUE LA TABLE D'ORIGINE, PEUT IMPORTE QUE LES DONNÉES SOIENT MODIFIÉES DEPUIS LA TABLE VIRTUELLE OU D'ORIGINE.

SQL

```
MariaDB [bibliotheque]> SELECT * FROM abonne ;
+-----+-----+
| id_abonne | prenom |
+-----+-----+
| 1 | Guillaume |
| 2 | Benoit |
| 3 | Chloe |
| 4 | Laura |
| 5 | alexandre |
+-----+
5 rows in set (0.00 sec)

MariaDB [bibliotheque]> UPDATE abonne SET prenom = 'Guillaumee' WHERE id_abonne = 1 ;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [bibliotheque]> SELECT * FROM abonne ;
+-----+-----+
| id_abonne | prenom |
+-----+-----+
| 1 | Guillaumee |
| 2 | Benoit |
| 3 | Chloe |
| 4 | Laura |
| 5 | alexandre |
+-----+
5 rows in set (0.00 sec)
```

SQL

```
MariaDB [bibliotheque]> SELECT * FROM abonne ;
+-----+-----+
| id_abonne | prenom   |
+-----+-----+
|      1    | Guillaumee |
|      2    | Benoit    |
|      3    | Chloe     |
|      4    | Laura     |
|      5    | alexandre |
+-----+-----+
5 rows in set (0.00 sec)

MariaDB [bibliotheque]> SELECT * FROM vue_emprunt ;
+-----+-----+-----+
| prenom   | date_sortie | titre        |
+-----+-----+-----+
| Guillaumee | 2016-12-07  | Une vie      |
| Benoit    | 2016-12-07  | Bel-Ami      |
| Chloe     | 2016-12-11  | Une vie      |
| Laura     | 2016-12-12  | Le Petit chose |
| Guillaumee | 2016-12-15  | La Reine Margot |
| Benoit    | 2017-01-02  | Les Trois Mousquetaires |
| Chloe     | 2017-02-15  | Les Trois Mousquetaires |
| Benoit    | 2017-02-20  | Une vie      |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

SQL

```
MariaDB [bibliotheque]> UPDATE vue_emprunt SET prenom = 'Guillaumeee' WHERE prenom = 'Guillaumee' ;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [bibliotheque]> SELECT * FROM vue_emprunt ;
+-----+-----+-----+
| prenom | date_sortie | titre |
+-----+-----+-----+
| Guillaumeee | 2016-12-07 | Une vie
| Benoit | 2016-12-07 | Bel Ami
| Chloe | 2016-12-11 | Une vie
| Laura | 2016-12-12 | Le Petit chose
| Guillaumeee | 2016-12-15 | La Reine Margot
| Benoit | 2017-01-02 | Les Trois Mousquetaires
| Chloe | 2017-02-15 | Les Trois Mousquetaires
| Benoit | 2017-02-20 | Une vie
+-----+-----+-----+
8 rows in set (0.00 sec)
```

SQL

```
MariaDB [bibliotheque]> SELECT * FROM abonne ;
+-----+-----+
| id_abonne | prenom |
+-----+-----+
|      1 | Guillaumeee |
|      2 | Benoit      |
|      3 | Chloe       |
|      4 | Laura        |
|      5 | alexandre   |
+-----+-----+
5 rows in set (0.00 sec)

MariaDB [bibliotheque]> ■
```

SQL

TABLE TEMPORAIRE

SQL

UNE TABLE TEMPORAIRE EST AUTOMATIQUEMENT SUPPRIMÉE À LA FERMETURE DE LA CONSOLE SQL.

C'EST UN COPIÉ COLLÉ DES DONNÉES MAIS ELLE NE SONT PAS LIÉES AVEC LA TABLE D'ORIGINE SI UNE MODIFICATION EST FAITE.

C'EST SEULEMENT UNE SAUVEGARDE DES DONNÉES.

CONTRAIEREMENT À LA TABLE VIRTUELLE, LA TABLE TEMPORAIRE N'EST PAS CONSTAMMENT À JOUR.

SQL

```
CREATE TEMPORARY TABLE emprunt2016 AS SELECT * FROM emprunt WHERE  
date_sortie BETWEEN '2016-01-01' AND '2016-12-31' ;
```

```
SELECT * FROM emprunt2016 ;
```

```
SHOW TABLES ;
```

```
T
```

SQL

```
MariaDB [bibliotheque]> CREATE TEMPORARY TABLE emprunt2016 AS SELECT * FROM emprunt WHERE date_sortie BETWEEN '2016-01-01' AND '2016-12-31' ;
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [bibliotheque]> SHOW TABLES ;
+-----+
| Tables_in_bibliotheque |
+-----+
| abonne
| emprunt
| livre
| vue_emprunt
+-----+
4 rows in set (0.00 sec)

MariaDB [bibliotheque]> SELECT * FROM emprunt2016 ;
+-----+-----+-----+-----+-----+
| id_emprunt | id_livre | id_abonne | date_sortie | date_rendu |
+-----+-----+-----+-----+-----+
|      1 |    100 |        1 | 2016-12-07 | 2016-12-11 |
|      2 |    101 |        2 | 2016-12-07 | 2016-12-18 |
|      3 |    100 |        3 | 2016-12-11 | 2016-12-19 |
|      4 |    103 |        4 | 2016-12-12 | 2016-12-22 |
|      5 |    104 |        1 | 2016-12-15 | 2016-12-30 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

SQL

VARIABLES

SQL

TRANSACTION

SQL

START TRANSACTION;

ROLLBACK

```
-- TRANSACTION
START TRANSACTION ;
SELECT * FROM employes ;
UPDATE employes SET salaire = 1000 ;
SELECT * FROM employes ;
ROLLBACK ;
-- ROLLBACK annule et TERMINE la transaction.
SELECT * FROM employes ;
```

ROLLBACK

```
-- TRANSACTION
START TRANSACTION ;
SELECT * FROM employes ;
UPDATE employes SET salaire = 1000 ;
SELECT * FROM employes ;
ROLLBACK ;
-- ROLLBACK annule et TERMINE la transaction.
COMMIT ;
-- COMMIT valide et TERMINE la transaction.
SELECT * FROM employes ;
```

ROLLBACK

```
-- TRANSACTION AVANCEE
START TRANSACTION ;
SELECT * FROM employes ;
SAVEPOINT point1 ;

UPDATE employes SET salaire = 3000 WHERE id_employes = 509 ;
SELECT * FROM employes ;
SAVEPOINT point2 ;

UPDATE employes SET salaire = 3100 WHERE id_employes = 509 ;
SELECT * FROM employes ;
SAVEPOINT point3 ;

UPDATE employes SET salaire = 3200 WHERE id_employes = 509 ;
SELECT * FROM employes ;
SAVEPOINT point4 ;

UPDATE employes SET salaire = 3300 WHERE id_employes = 509 ;
SELECT * FROM employes ;
SAVEPOINT point5 ;

ROLLBACK TO point4 ;
SELECT * FROM employes ;
```

ROLLBACK

```
SAVEPOINT point1 ;  
  
UPDATE employes SET salaire = 3000 WHERE id_employes = 509 ;  
SELECT * FROM employes ;  
SAVEPOINT point2 ;  
  
UPDATE employes SET salaire = 3100 WHERE id_employes = 509 ;  
SELECT * FROM employes ;  
SAVEPOINT point3 ;  
  
UPDATE employes SET salaire = 3200 WHERE id_employes = 509 ;  
SELECT * FROM employes ;  
SAVEPOINT point4 ;  
  
UPDATE employes SET salaire = 3300 WHERE id_employes = 509 ;  
SELECT * FROM employes ;  
SAVEPOINT point5 ;  
  
ROLLBACK TO point4 ;  
SELECT * FROM employes ;  
  
ROLLBACK TO point5 ; -- /\ erreur.  
  
ROLLBACK TO point3 ;  
SELECT * FROM employes ;|
```

SQL

REQUÊTE PRÉPARÉE

SQL

ANALYSE, INTERPRÉTATION ET EXÉCUTION;

SQL

```
-- Requête Préparée

SELECT * FROM employes WHERE service = 'commercial';

-- Analyse, Interprétation, Exécution
-- 15 étapes
```

SQL

EN PRÉPARANT LA REQUÊTE NOUS ÉVITONS LA RÉPÉTITION DES ÉTAPES D'ANALYSE ET D'INTERPRÉTATION.

UNE FOIS PRÉPARÉE SEULE L'ÉTAPE D'EXÉCUTION EST LANCÉE.

SQL

```
-- Req Préparée
PREPARE req FROM 'SELECT * FROM employes WHERE service = ?';

SET @service = 'commercial' ;
SELECT @service ;

EXECUTE req USING @service ;
```

SQL

```
-- Req Préparée
PREPARE req FROM 'SELECT * FROM employes WHERE service = ?';
-- Analyse, Interprétation

SET @service = 'commercial' ;
SELECT @service ;| I

EXECUTE req USING @service ; -- Exécution
-- 7 étapes
```

SQL

LA VARIABLE PEUT ÊTRE REDÉFINIT

SQL

```
SET @service = 'commercial' ;
SELECT @service ;

EXECUTE req USING @service ; -- Exécution
EXECUTE req USING @service ; -- Exécution
EXECUTE req USING @service ; -- Exécution

SET @service = 'informatique' ;
SELECT @service ;

EXECUTE req USING @service ; -- Exécution
EXECUTE req USING @service ; -- Exécution
-- 7 étapes
```

SQL

PHP MY ADMIN

SQL

CONTRAINTE D'INTÉGRITÉ

SQL

IMPORT / EXPORT