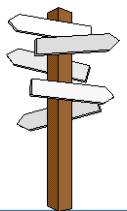


SQL

Plus d'informations sur <http://www.dawan.fr>
Contactez notre service commercial au [09.72.37.73.73](tel:09.72.37.73.73) (prix d'un appel local)

Objectifs

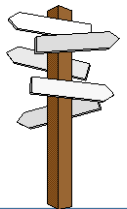
- Découvrir SQL
- Savoir modéliser une Base de Données
- Être capable de créer les requêtes standards de sélection, d'enregistrement, de modification ou de suppression de données
- Savoir relier des tables entre elles



Plan de l'intervention



- Introduction
- Commandes simples
- Présentation des SGBDr
- Utilisation avancée



Introduction

Introduction

Bases de données

- Une base de données ?
- Un SGBD ?

Systeme de Gestion de Bases de Données

- Le SQL ?

Structured Query Language

Base de Données



- Une base de données est une collection de données organisées de façon à être facilement accessibles, administrées et mises à jour.
- Les bases de données peuvent être classées par le type de contenu qu'elles renferment : bibliographique, full text, images ou des nombres....

SGBD



- Un système de gestion de base de données (SGBD) est une application qui sert comme son nom l'indique à stocker et accéder à des données.
- **MySQL** : Sous licence GPL (gratuite) performante quoique légèrement incomplète
- **PostgreSQL** : Également Open Source performante et offrant de nombreuses fonctionnalités
- **Oracle** : La base de données professionnelle
- **DB2** (IBM): Une autre base de données professionnelle
- **SQL Server** (MS): La solution proposée par Microsoft

SQL(Structured Query Language)



- langage de requête structurée
- langage de définition de données (LDD)
- langage de manipulation des données qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.
- langage de contrôle de données (LCD)
- définir des permissions au niveau des utilisateurs d'une base de données.

Introduction



Historique, versions et normalisation

1970 – 1980 : apparition du langage SQL et utilisation par IBM puis Oracle

1985 : SQL devient une norme ISO internationale

1992 : SQL 2 évolution majeure <https://en.wikipedia.org/wiki/SQL-92>

1999 : SQL 3 , supporte bien sûr la norme précédente SQL2 (compatibilité ascendante) mais propose également des extensions objets.

Méthode MERISE

1. Dictionnaire de Données
2. Modèle Conceptuel de Données
3. Modèle Logique de Données

Méthode Merise

Dictionnaire de Données

- regroupe les données que vous aurez à conserver dans votre base (et qui figureront donc dans le MCD).

Le Dictionnaire de Données contient :

- Code mnémonique : Ex designation_produit, nom_produit ...etc
- Type de donnée : type caractère, numérique, boolean, date ...etc
- Taille des données : ex varchar(255), int(4) ...etc

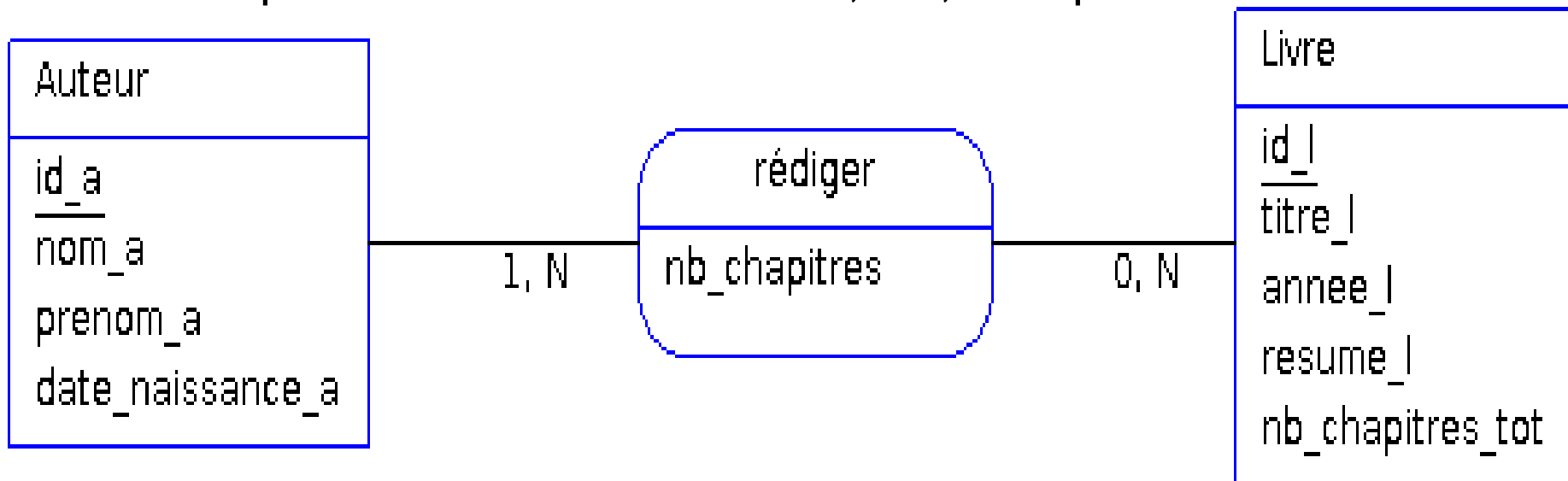
Méthode Merise

- **Modèle Conceptuel de Données (MCD)**
- Définir les entités et relations de notre BD
- Entité : ensemble de propriétés encore appelées attributs ou caractéristiques
- Associations : Une association définit un lien sémantique entre une ou plusieurs entités

Méthode Merise

Modèle Conceptuel de Données (MCD)

- Exemple d'Entités et d'Association :
- Remarquez bien les cardinalités 0,N/1,N de part et d'autre des entités



Les cardinalités indiquées ici signifient qu'un auteur rédige un ou plusieurs livre (mais au moins un). On peut avoir des livres sans auteur.

Avec les cardinalités inversées cela signifierait que les livres ont au moins un auteur, et qu'un auteur peut ne pas avoir rédigé de livre

Pour chaque livre, on connaît le nombre de chapitres rédigés par l'auteur.

Modèle Conceptuel de Données (MCD)

- **0,1** : occurrence du type entité peut exister tout en n'étant impliquée dans aucune association et peut être impliquée dans au maximum une association.
- **0,n** : c'est la cardinalité la plus ouverte ; une occurrence du type entité peut exister tout en n'étant impliquée dans aucune association et peut être impliquée, sans limitation, dans plusieurs associations.
- **1,1** : une occurrence du type entité ne peut exister que si elle est impliquée dans exactement (au moins et au plus) une association.
- **1,n** : une occurrence du type entité ne peut exister que si elle est impliquée dans au moins une association.

Méthode Merise

Modèle Logique de Données (MLD)

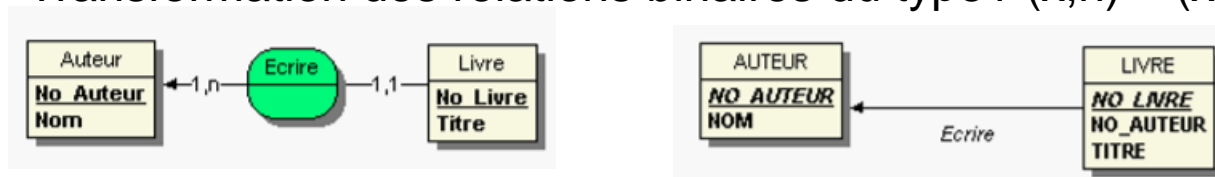
- Etape pour établir une modélisation des données au niveau logique (ou relationnel) à partir du MCD, puis comment passer à l'étape de création des tables
- Cette étape suit des Règles de conversion

Méthode Merise

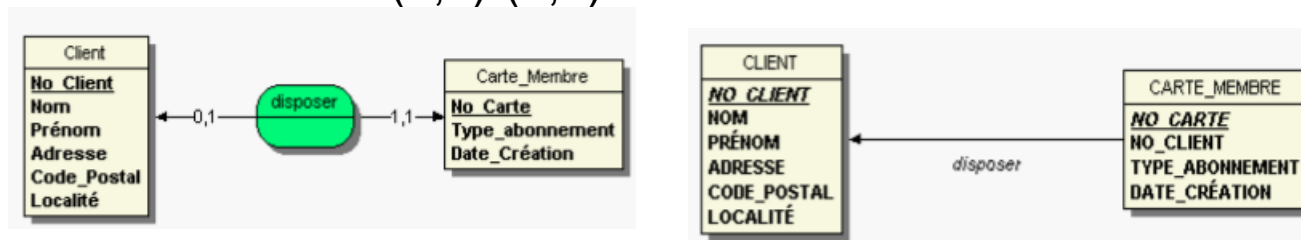
Modèle Logique de Données (MLD)

- Cette étape suit des Règles de conversion

Transformation des relations binaires du type $1(x,n) - (x,1)$



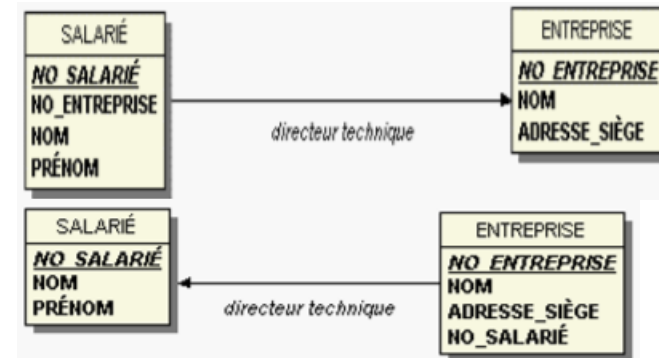
Relation binaire $(0,1)-(1,1)$



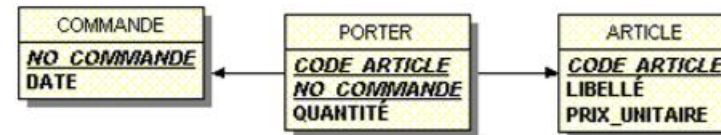
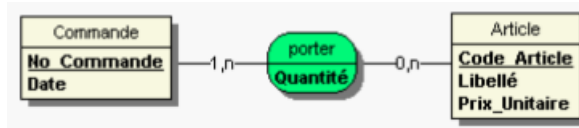
Méthode Merise

Modèle Logique de Données (MLD) Règles de conversion

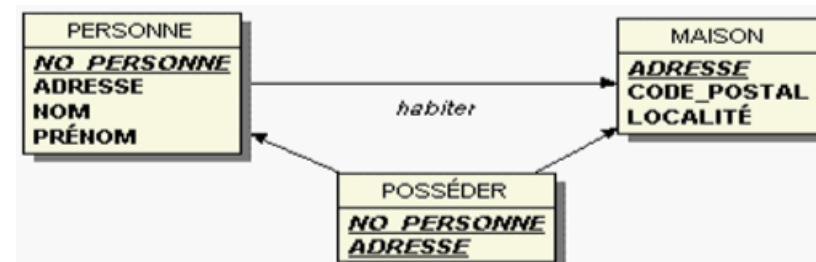
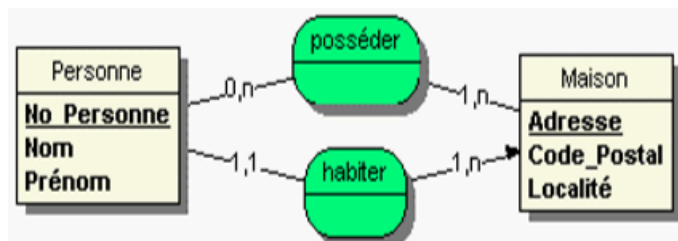
Relation binaire (0,1)-(0,1) (2 solutions)



Transformation des relations binaires du type (x,n) – (x,n)



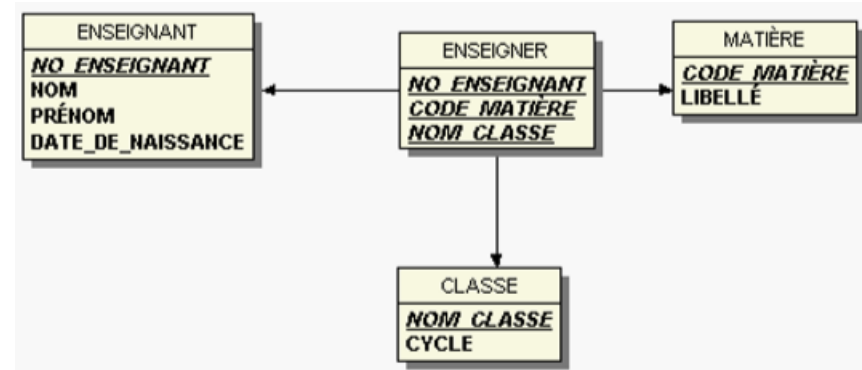
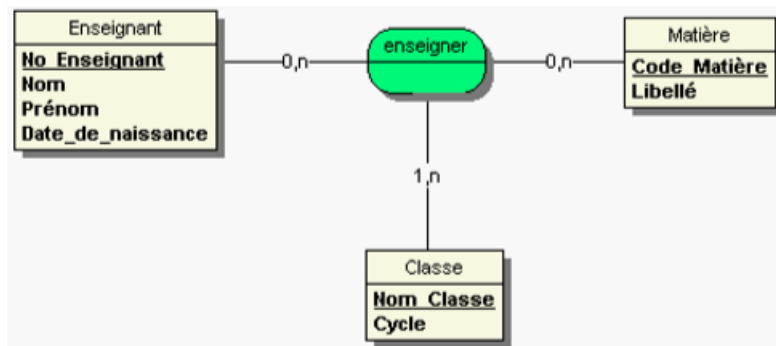
Transformation de plusieurs relations entre 2 entités



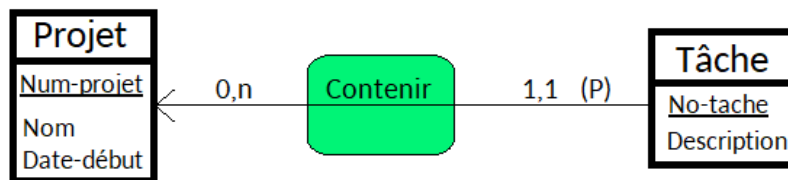
Méthode Merise

Modèle Logique de Données (MLD) Règles de conversion

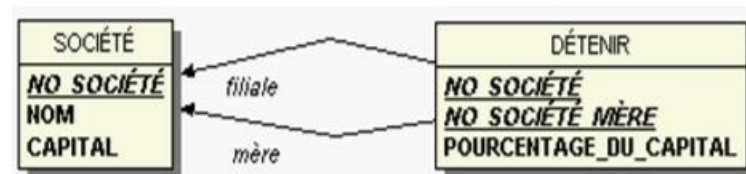
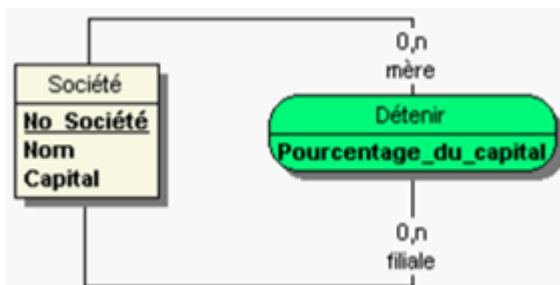
Transformation des relations ternaires



Transformation de l'identifiant relatif



Transformation des relations réflexives



Formes normales

Le but est d'améliorer la structure de la DB, éviter les redondances et autre problème de conception.

1NF (1^{ère} forme normale) : un champ ne peut pas avoir plusieurs valeurs

propriétaire	modèle de voiture
Jean Dupont	Peugeot 106, Renault twingo
Fred Kim	Bmw Z1
Pedro Ama	Peugeot 106, Bmw Z1

propriétaire	modèle de voiture
Jean Dupont	Peugeot 106
Jean Dupont	Renault twingo
Fred Kim	Bmw Z1
Pedro Ama	Peugeot 106
Pedro Ama	Bmw Z1

2NF : un champ ne doit pas dépendre d'une partie de la clef

<u>nom</u>	<u>prénom</u>	ville	jour de la fete

<u>nom</u>	<u>prénom</u>	ville

<u>prénom</u>	jour de la fete

Il faut séparer en deux tables, le jour de la fête dépend du prénom uniquement, pas du nom

Forme normale

3NF : un champ ne doit pas dépendre d'un autre champ non clef

~~| <u>nom</u> | <u>prénom</u> | ville | nombre habitant |
|------------|---------------|-------|-----------------|
| | | | |
| | | | |~~

<u>nom</u>	<u>prénom</u>	ville

<u>ville</u>	nombre habitant

Ici le nombre d'habitant ne dépend pas de la clef nom, prénom

Forme normale de Boyce-Codd un champ non clef ne peut pas déterminer un autre champ

~~| <u>nom</u> | <u>pays</u> | ville |
|------------|-------------|-------|
| | | |
| | | |~~

<u>nom</u>	<u>ville</u>

<u>ville</u>	pays

Il existe d'autres formes normales 4NF,... <https://stph.scenari-community.org/bdd/nor1-lin/co/norAC03.html>

Exercice MCD MLD

- <http://pise.info/modelisation/enonces.htm>
- Exercice créer les tables pour gérer les candidat faisant un QCM et avoir leur résultats.
- Modélisation de commandes (produit, commandes, détails)

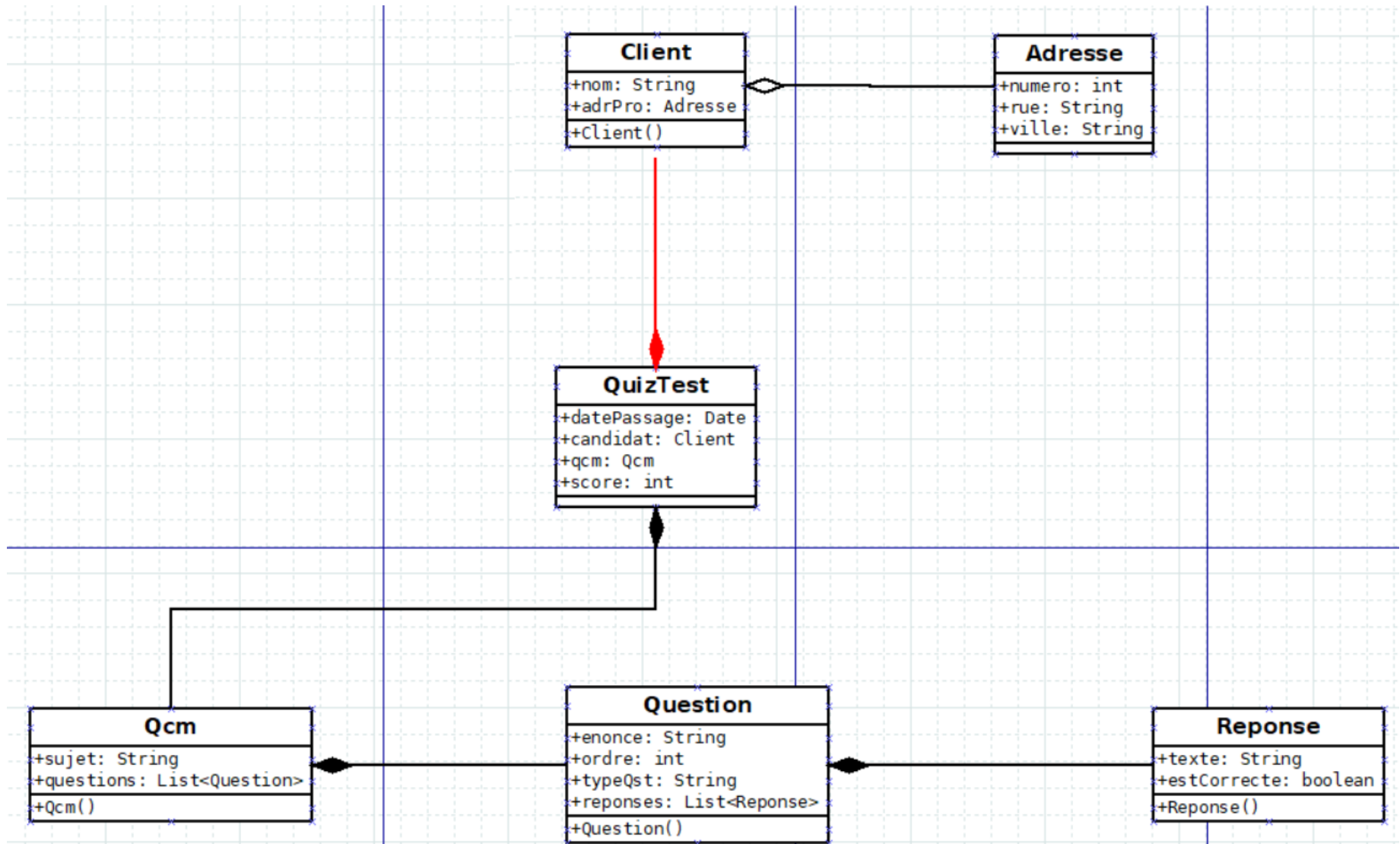
On peut utiliser MysqlWorkBench pour la modélisation une fois qu'on sait faire sur papier *(MysqlWorkBench permet également de faire du reverse engineering a partir de fichiers SQL – mais on verra cela plus tard)*

Pour info : Outils de modélisation MCD – MLD

Il existe de nombreux outils comme MysqlWorkbench et qui permettent en plus de gérer l'ensemble des schéma UML et dont on peut se servir pour créer notre MCD et MLD : DIA, Umbrello,

Mais l'UML est un vaste sujet dont il n'est pas question dans cette formation et qui permet en outre de concevoir une base de données.

Nb en UML les notations de cardinalités sont inversées (look accross notation)



Commandes simples

CREATE DATABASE NomDeLaBase;

SHOW DATABASES;

USE NomDeLaBase;

SHOW TABLES;

L'interclassement (*utf8_general_ci, etc..*) définit le type de données stockées mais également la manière dont va se faire le classement des données et donc le résultat des recherches.

_ci signifie case insensitive (=non sensible à la casse), les recherches de 'c' ou 'C' donnent la même chose). Pour stocker les caractères de la plupart des langues on choisit *utf8_xxx*

Commandes simples



Les types de données

- Numériques : TINYINT / SMALLINT / INTEGER
FLOAT / DOUBLE (UNSIGNED)
- Textuels : CHAR / VARCHAR / TEXT / LONGTEXT
- Temporels : DATE / TIME / DATETIME / TIMESTAMP
- Il faut parfois indiquer la taille des valeurs stockées notamment pour les CHAR et les VARCHAR, mais pas nécessairement pour les INT (meme si on peut..
<https://stackoverflow.com/questions/5634104/what-is-the-size-of-column-of-int11-in-mysql-in-bytes>)

Commandes simples

Les types de données

Avec FIREFOX, (pas edge) , quand on crée les champs sur PHPMyAdmin, la taille des TYPE est indiquée lors du choix du type dans le menu déroulant

Nom	Type ?	Taille/Valeurs* ?	Valeur par défaut ?	Interclassement
name	VARCHAR	250	Aucun(e)	utf8_unicode_c
	INT			
	VARCHAR			
	TEXT			

Un nombre entier de 4 octets. La fourchette des entiers relatifs est de -2 147 483 648 à 2 147 483 647. Pour les entiers positifs, c'est de 0 à 4 294 967 295

Commandes simples

Créations d'une table

CREATE TABLE [*nomTable*] (
 [*nomChamp*] [*type*] [NULL] [*options*],
 PRIMARY KEY ([*nomChamp*]));

NULL : champ facultatif

NOT NULL : champ obligatoire

Options : AUTO_INCREMENT, DEFAULT, UNIQUE

Commandes simples



Modifications de structure de la table

ALTER TABLE [*nomTable*] ...

MODIFY [*nomChamp*] [type] [NULL] [*options*];

ADD COLUMN [*nomChamp*] [type] [NULL] [*options*];

DROP COLUMN [*nomChamp*];

DROP TABLE [*nomTable*];

Commandes simples



Insertion de données

INSERT INTO [*nomTable*] (*champX*, ...)
VALUES (*valeurX*, ...);

Insertion sur plusieurs lignes possible.

Commandes simples



Insertion de données

Colonne muette :

- Colonne clé avec auto_increment > numéro automatique inséré
- Colonne avec valeur par défaut > valeur par défaut insérée
- Colonne *nullable* > valeur NULL insérée
- Colonne sans valeur par défaut et non nullable
 - > l'insertion de la ligne est refusée

Commandes simples



Modifier les données

Modifier :

UPDATE *nomTable*

SET *champX = valeurX*

WHERE *id = 1;*

Pas de condition : met à jour toute la table !

Commandes simples



Suppression de données

Supprimer :

```
DELETE FROM nomTable  
WHERE id = 1;
```

Pas de condition : met à jour toute la table !

Vider la table :

```
TRUNCATE TABLE nomTable ;
```


Commandes simples



Accéder aux données

•
Sélectionner :

```
SELECT * FROM nomTable;
```

Opération et Alias de colonne :

```
SELECT prix_ht, prix_ht*1.20 AS prix_ttc  
FROM nomTable ;
```

Commandes simples



Accéder aux données

Clause WHERE :

SELECT *champX, champY* **FROM** *nomTable*
WHERE *condition1*
AND *condition2...* ;

Comparaison : =, <, <=, >, >=, <> / !=, LIKE, BETWEEN

Commandes simples



Accéder aux données

Clause ORDER BY :

Organiser les résultats (alphabétique, numérique)

Clause LIMIT :

Limiter le nombre de résultats.

Commandes simples



Accéder aux données (à noter)

De meme que l'on a des alias pour les champs on a également des alias pour les tables

Alias de table :

SELECT *nT.champX* as *cX*, *champY* as *cY*

FROM *nomTable* **AS** *nT*

WHERE *nT.champX* = 1 **AND** *ChampY* = 2;

ORDER BY *cY*

On ne peut plus utiliser *nomTable* une fois qu'elle est nommée avec un Alias

Les alias de champs ne peuvent pas être repris dans le WHERE.

Alias de table : utile pour les jointures d'une table avec elle-même (on verra cela plus tard)

Commandes simples



L'agrégation

- **SUM** calcul la somme de la colonne
- **AVG** calcul la moyenne de la colonne
- **MAX** calcul le maximum de la colonne
- **MIN** calcul le minimum de la colonne
- **COUNT** dénombre les éléments
- *Uniquement dans le SELECT*

Commandes simples



Autres fonctions

- NOW() -> heure actuelle
- CONCAT('bonjour', '-', 'toto') → 'bonjour-toto'
- REPLACE('hello toto', 'hello', 'bonjour') ->'bonjour toto'
- SUBSTRING()

Commandes simples



Le regroupement

Clause **GROUP BY** :

Regrouper les calculs par valeur d'une colonne

```
SELECT id_genre, COUNT(id_livre) AS nb_livre  
FROM livre  
GROUP BY id_genre ;
```

Commandes simples



L'ordre des mots clés SQL

SELECT...

FROM...

JOIN...

WHERE...

AND/OR...

GROUP BY...

HAVING...

ORDER BY...

LIMIT...

Commandes avancées

Les Jointures : Requêtes multi-tables

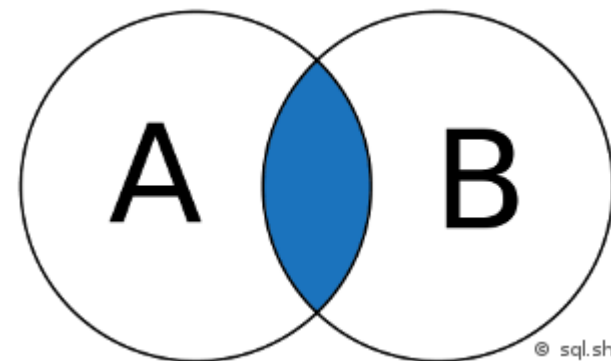
UPDATE avec jointures

Commandes avancées

Jointures : Requêtes multi-tables

Les Jointures :

- **INNER JOIN**



```
SELECT * FROM A
```

```
INNER JOIN B ON A.key = B.key
```

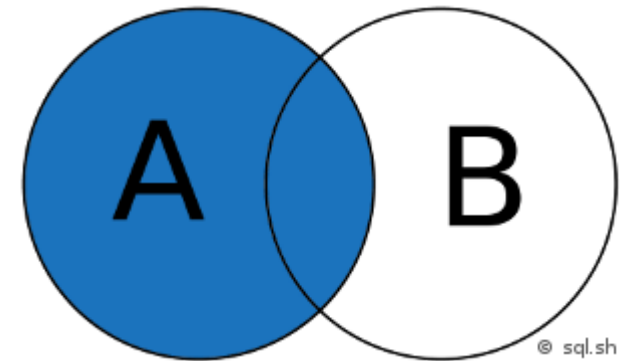
Commandes avancées

Jointures : Requêtes multi-tables

- **LEFT JOIN**

```
SELECT * FROM A
```

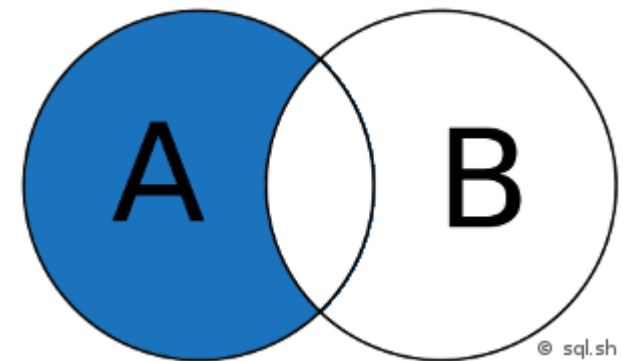
```
LEFT JOIN B ON A.key = B.key ;
```



```
SELECT * FROM A
```

```
LEFT JOIN B ON A.key = B.key
```

```
WHERE B.key IS NULL ;
```



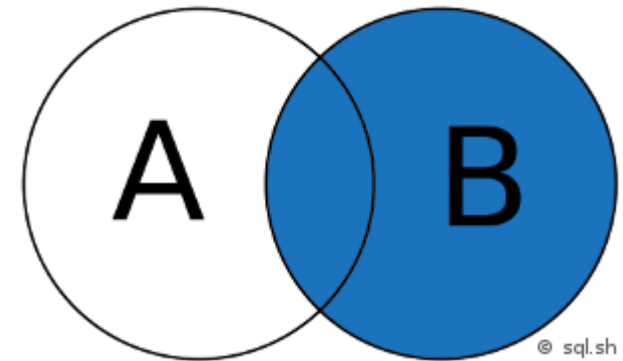
Commandes avancées

Jointures : Requêtes multi-tables

- **RIGHT JOIN**

SELECT * FROM A

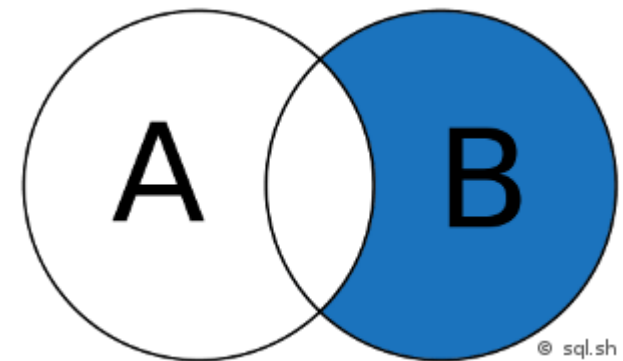
RIGHT JOIN B ON A.key = B.key ;



SELECT * FROM A

RIGHT JOIN B ON A.key = B.key

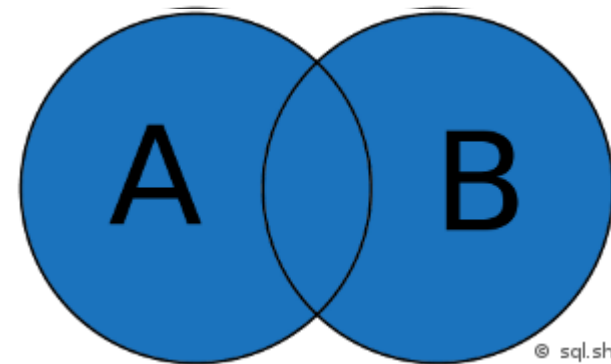
WHERE A.key IS NULL ;



Commandes avancées

Jointures : Requêtes multi-tables

- **FULL JOIN**



```
SELECT * FROM A
```

```
FULL JOIN B ON A.key = B.key
```

Présentation des SGBDr

Bases de Données Relationnelles

- **Base de Données avec relations :**

*Une table est liée à une seconde table, ou plus, par une relation de **clé étrangère**, suivant des contraintes strictes dites « d'intégrité référentielle ».*

Définition de la clé étrangère

Définition des clés :

ALTER TABLE [nomTable1] ADD INDEX (id_table1); /* Uniquement MySQL */

ALTER TABLE [nomTable1]

ADD CONSTRAINT nomDeLaContrainte (ex
FK_table_name_nomColonne)

FOREIGN KEY nomTable1 (id_table1)

REFERENCES [nomTable2] (id_table2) ;

Contraintes d'intégrité référentielle

Définition des actions en chaîne :

... ON UPDATE [*value*] ON DELETE [*value*]

**CASCADE / SET NULL / SET DEFAULT / RESTRICT
/ NO ACTION**

Utilisation des relations

Sélection de données sur plusieurs tables

```
SELECT *
```

```
FROM tableA, tableB, tableC
```

```
WHERE tableA.id_tabB = tableB.id
```

```
AND tableA.id_tabC = tableC.id;
```

☐ *Requête INNER JOIN*

Utilisation avancée

Les requêtes imbriquées

Traiter une requête en fonction d'une autre :

```
SELECT * FROM genre  
WHERE id_genre [NOT] IN (  
    SELECT DISTINCT id_genre  
    FROM livre  
) ;
```

Les vues : créer / utiliser

Requêtes préenregistrées sur la base de données, que l'on qualifie aussi de **tables virtuelles**.

```
CREATE VIEW view_nomDeLaVue AS  
SELECT .... ;
```

```
SELECT * FROM nomDeLaVue ;
```

Les vues : modifier / supprimer

- **Modifier :**

```
ALTER VIEW nomDeLaVue AS  
SELECT .... ;
```

- **Supprimer :**

```
DROP VIEW nomDeLaVue ;
```

Les vues : limites

- **Toujours une requête de sélection**
- **Une vue : une requête (possibilité d'imbriquer)**

Les procédures stockées (*approche*)

- *Ensemble d'instruction de manipulation de données (**INSERT**, **UPDATE**, **DELETE**) qui peuvent être exécutés par un simple appel.*
- *L'appel diffère d'un SGBD à l'autre :*
EXEC (avec SQL Server), **CALL** (avec MySQL), etc...

Les procédures stockées (*approche*)

- **Exemple de procédure stockée :**

```
CREATE PROCEDURE nomProcedure()
```

```
BEGIN
```

```
    ...Requête SQL
```

```
END;
```

Utilisation avancée

Exemple d'une procédure stockée pour le changement du nom du client ayant l'ID = 4, avec un paramètre - avec PHPMyAdmin

✓ La procédure `change_client4` a été modifiée.

```

DROP PROCEDURE `proc2`; CREATE DEFINER=`root`@`localhost` PROCEDURE `change_client4`(IN `new_nom` VARCHAR(50)) NOT DETERMINISTIC CONTAINS SQL SQL SECURITY DEFINER BEGIN UPDATE customers SET customerName = new_nom WHERE CustomerID = 4; END
    
```

Procédures stockées

Nom	Action
proc1	Éditer Exécuter Exporter Supprimer
change_client4	Éditer Exécuter Exporter Supprimer

Exporter la procédure `change_client4`

```

1 DELIMITER $$
2 CREATE DEFINER=`root`@`localhost` PROCEDURE
3 `change_client4`(IN `new_nom` VARCHAR(50))
4 BEGIN
5 UPDATE customers SET customerName = new_nom WHERE
6 CustomerID = 4;
7 END$$
8 DELIMITER ;
    
```

Éditer une procédure

Détails

Nom de la procédure: change_client4

Type: PROCEDURE

Direction	Nom	Type	Taille/Valeurs*	Options
IN	new_nom	VARCHAR	50	Jeu de c

Ajouter un paramètre

Définition

```

1 BEGIN
2
3 UPDATE customers SET customerName = new_nom WHERE CustomerID = 4;
4
5 END
    
```

L'appel se fait en cliquant ici ou avec CALL

Exécuter une ou des requêtes SQL sur la table « test2.customers »:

```

1 CALL change_client4('toto');
    
```

✓ Affichage des lignes 0 - 0 (total de 1, traitement en 0,0000 seconde(s).)

```

SELECT * FROM `customers` WHERE CustomerID = 4
    
```

Options	CustomerID	CustomerName	ContactName	Address
Éditer Copier Supprimer	4	toto	Thomas Hardy	120 Hanove

Utilisation avancée



Les fonctions doivent renvoyer une valeur
(pas obligatoire pour les procédures)

```
delimiter $$  
CREATE FUNCTION mafonction(x INT) RETURNS INT  
BEGIN  
    DECLARE sortie INT;  
    SET sortie = x*2;  
    RETURN (sortie);  
END$$
```

Les fonctions vont plutôt servir dans un SELECT ou un UPDATE
comme la fonction CONCAT ou ROUND – **PAS** d'appel avec un CALL

Pour plus d'explication sur la syntaxe voir slide suivante

Utilisation avancée



Avec l'interface de PHPMyAdmin, on peut créer une fonction : c'est comme pour une procédure, mais il faut changer le Type

Il n'est pas nécessaire de rédiger l'ensemble du script SQL de déclaration car l'interface nous simplifie le travail.

L'interface rajoute en plus des options supplémentaire comme le Créateur de la fonction, etc...

Il faut aller consulter <https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html>

The screenshot shows the 'Éditer une procédure' (Edit a procedure) interface in PHPMyAdmin. The 'Détails' tab is active. The 'Nom de la procédure' (Procedure name) is 'mon_addition'. The 'Type' is set to 'FUNCTION'. Below this, there is a table for 'Paramètres' (Parameters) with two rows: 'a' and 'b', both of type 'INT'. Each row has a 'Supprimer' (Delete) button. Below the parameters table is a button 'Ajouter un paramètre' (Add a parameter). The 'Type de retour' (Return type) is set to 'INT'. The 'Longueur/valeurs de retour' (Return length/values) is set to '11'. The 'Options de retour' (Return options) are empty. At the bottom, the 'Définition' (Definition) tab shows the SQL code for the function:

```
1 BEGIN
2 DECLARE res INT(4);
3 SET res = a+b;
4 RETURN (res);
5 END
```

Utilisation avancée



En faisant Exporter on récupère le code SQL entier de création de la fonction.

The screenshot shows the MySQL Workbench interface. On the left, a tree view shows a database named 'test2' containing 'Fonctions', 'Procédures', and 'Tables'. Under 'Fonctions', there is 'mon_addition'. A red arrow points from 'mon_addition' to the 'Export' button in the 'Action' column of the table grid. The table grid has columns: Nom, Action, Type, and Retourne. The row for 'mon_addition' shows 'mon_addition' in the 'Nom' column, 'Éditer', 'Exécuter', 'Exporter', and 'Supprimer' in the 'Action' column, 'FUNCTION' in the 'Type' column, and 'int(11)' in the 'Retourne' column. Below the table grid, there is a 'Nouvelle procédure' section with an 'Ajouter une procédure' button. To the right, a panel titled 'Exporter la procédure `mon_addition`' displays the SQL code for creating the function.

```
1 DELIMITER $$
2 CREATE DEFINER=`root`@`localhost` FUNCTION
3 `mon_addition`(`a` INT, `b` INT) RETURNS int(11)
4 NO SQL
5 BEGIN
6 DECLARE res INT(4);
7 SET res = a+b;
8 RETURN (res);
9 END$$
10 DELIMITER ;
```

Le délimiteur sert à changer le symbole d'exécution qui est « ; » afin que le code de la fonction ne soit pas exécuté pendant la déclaration de la fonction. On exécute que la déclaration de la fonction une fois que l'on a atteint le END. On met \$\$ ou // généralement pour remplacer temporairement le « ; »

Il y a d'autres éléments qui sont venus se rajouter DEFINER, NO SQL ... il faut aller voir la documentation :

<https://stackoverflow.com/questions/5634104/what-is-the-size-of-column-of-int11-in-mysql-in-bytes> DAWAN - Reproduction interdite sans autorisation

Utilisation avancée

Autre Exemple de script SAL avec déclaration de deux 2 fonctions et insertion d'un enregistrement dans une table, montrant l'utilisation du délimiteur

```
1  -- Commentaire
2  # Commentaire
3  /* Commentaire sur
4  plusieurs lignes */
5
6  DELIMITER $$ # Change le symbole ";" pour la fin d'instruction en "$$"
7  /* en effet on ne doit pas exécuter les les instructions maintenant mais quand la fonction sera appelée. */
8
9  # on n'est pas obligé
10 CREATE DEFINER=`root`@`localhost` FUNCTION `mon_addition`(`a` INT, `b` INT) RETURNS int(11)
11     NO SQL
12 BEGIN
13 DECLARE res INT4;
14 SET res = a+b;
15 RETURN (res);
16 END $$
17
18 #INSERT INTO `client` (`id`, `nom`) VALUES (NULL, 'XX'); #ne marchera pas car le délimiteur est '$$' et non plus ';'
19 #si la ligne précédente est exécutée avec un ';' à la fin elle fait planter le script et le script s'arrete là. Ce qui
20 a été exécuté avant n'est pas annulé.
21
22 INSERT INTO `client` (`id`, `nom`) VALUES (NULL, 'YY')$$ #sera exécuté normalement
23
24 CREATE DEFINER=`root`@`localhost` FUNCTION `ma_multiplication`(`a` INT, `b` INT) RETURNS int(11)
25     NO SQL
26 BEGIN
27 DECLARE res INT4;
28 SET res = a*b;
29 RETURN (res);
30 END $$
31
32 # on remet le delimiteur avec sa valeur 'habituelle'
33 DELIMITER ; #Avec mysql et phpmyadmin le délimiteur revient à ";" tout seul, mais on peut toujours l'indiquer.
```

Utilisation avancée

Appel de ma fonction 'mon_addition' dans un SELECT

```
SELECT prenom, age, mon_addition(age,10) as age_dans_dix_ans FROM `client`
```

+ Options

prenom	age	age_dans_dix_ans
Jeannie julie	51	61
tutu	NULL	NULL
Sophie cecile	50	60
Beatrice	NULL	NULL

On n'aurait pas pu faire CALL mon_addition(4,5)

Erreur

Requête SQL :

```
CALL mon_addition(25,10)
```

MySQL a répondu : ?

```
#1305 - PROCEDURE test2.mon_addition does not exist
```

Utilisation avancée



Il existe également :

*** des tâches planifiées qui se lance en fonction d'une date**

```
CREATE EVENT myevent ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
DO
UPDATE myschema.mytable SET mycol = mycol + 1;
https://dev.mysql.com/doc/refman/5.7/en/create-event.html
```

*** des déclencheurs qui se lance si une action est effectuée**
(lors de l'insertion dans une table particulière)

```
CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));

CREATE TRIGGER ins_sum BEFORE INSERT ON account
FOR EACH ROW SET @sum = @sum + NEW.amount;

SET @sum = 0;
INSERT INTO account VALUES(137, 14.98), (141, 1937.50), (97, -100.00);
SELECT @sum AS 'Total amount inserted'
```


Les optimisations

- **Vérifier la charges des requêtes**
- **Sauvegarder les données ? Archiver ?**

Utilisation avancée

Les optimisations

L'outil SQL Explain

EXPLAIN select * from nom_table

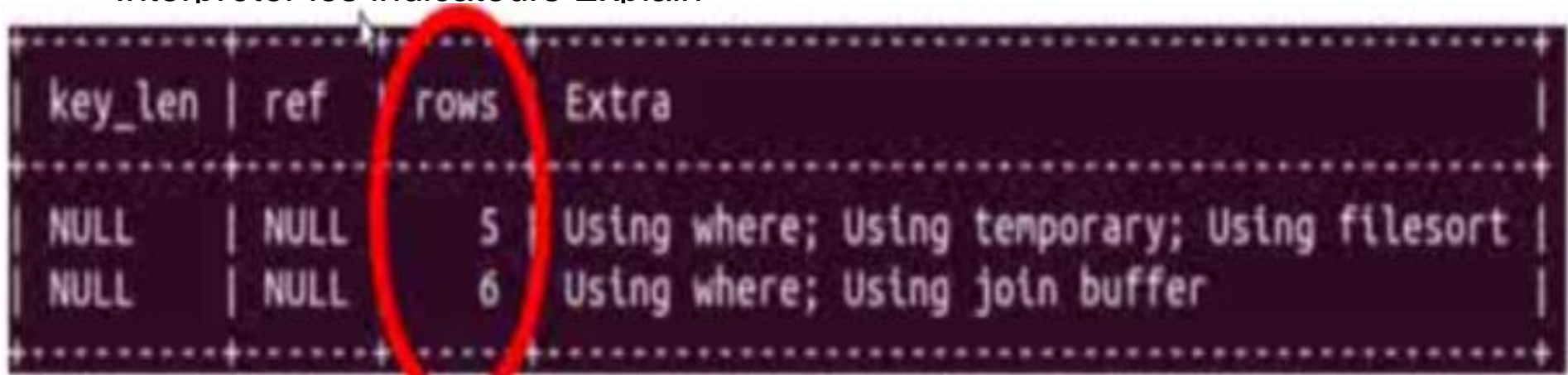
id	select_type	table	type	possible_keys	key
1	SIMPLE	instructor	ALL	NULL	NULL
1	SIMPLE	grade	ALL	NULL	NULL

key_len	ref	rows	Extra
NULL	NULL	5	Using where; Using temporary; Using filesort
NULL	NULL	6	Using where; Using join buffer

Utilisation avancée

Les optimisations

- Charge de la requête SQL
- Interpréter les indicateurs Explain

A screenshot of a MySQL EXPLAIN output. The 'rows' column is circled in red. The output shows two rows of data, both with NULL values for key_len and ref, and different row counts and extra information.

key_len	ref	rows	Extra
NULL	NULL	5	Using where; Using temporary; Using filesort
NULL	NULL	6	Using where; Using join buffer

Sauvegarder les données

- `mysqldump --user=mon_user --password=mon_password --all-databases > ALLBD_Backup.sql`
- `mysqldump --user=mon_user --password=mon_password --databases nom_de_la_base > BD_name_Backup.sql`

Restaurez les données

- `mysql --user=user_name --password=password < file_Backup.sql`
- `mysql --user=user_name --password=password database_name < file_Backup.sql`

Archivez les données

- `mysqldump <commandes> | gzip > archive_file_name.sql.gz`

SQL



Plus d'informations sur <http://www.dawan.fr>
Contactez notre service commercial au **0800.10.10.97** (prix d'un appel local)