SQL

More informations on http://www.dawan.fr

Objectives

- ✓ Discover SQL
- ✓ Know how to modelize a database
- ✓ Being able to create standard queries to select, update, insert and delete data
- ✓ Learn how to link tables with constraints

Curriculum

- ✓ Introduction
- ✓ Simple queries
- ✓ Introduction to RDBMS
- ✓ Advanced usages

Introduction

Introduction

Database

✓ What is a database?

1) A database is a persisted collection of data organised to be easily accessed, managed and updated.

2) Databases can be sorted by the type of thier content : bibliography, full text, images or numbers ….

Database

- ✓ What is a DBMS ?
- ✓ *Database Management System*

- ✓ SQL ?
- ✓ *Structured Query Language*

Introduction

History, versions and evolution

1970 : First query langage by IBM.

1982-1989 : SQL 1.

1992 : SQL 2 : Based upon SEQUEL from IBM .

1999 : SQL 3 : SQL3 is an update dromSQL2. It allows extensions for objects, and set relationnals constraints between tables,

* 4 complexity levels : Entry ,Transitional, Intermediate and Full

DBMS

- A database management system **(DBMS)** is a software which only goals is to **store** and **access** data.
- **MySQL** : Under GNU licence (free) most commonly adopted. Fast and light.
- **PostgreSQL** : (PSQL) Also Open Source. High performance, with a lot of tools for customization.
- **Oracle** : Professionnal DBMS
- **DB2** (IBM): Old but still used
- **SQL Server** (MS): DBMS from Microsoft

Introduction

MERISE method is a designing method to develop and realize data model of software projects.

1. Conceptual Data Model
2. Logical Data Model
3. Physical Data Model

Merise Method

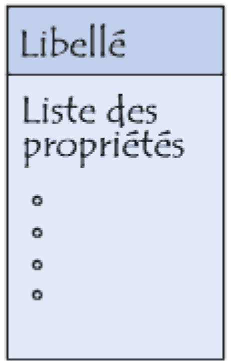
1. Conceptual Data Model (CDM)

✓ The CDM is a way to normalize the desing of data model. It is a representation of the datas and their interactions.

✓ It allows to define the entities and the relations of our database.

a) Entity : a gathering of defining properties

b) Relations : a relation defines a semantic link between one or two entities



Structural queries

Commands

CREATE DATABASE DBName;

SHOW DATABASES;

USE DBName;

SHOW TABLES;

Table Commands

Data types

- ✓ Number : TINYINT (1 Octets) / SMALLINT (2 octets) / INTEGER (4 octets)/ FLOAT / DOUBLE (UNSIGNED)

- ✓ Text : CHAR / VARCHAR / TEXT / LONGTEXT
- ✓ Time : DATE / TIME / DATETIME / TIMESTAMP

Table Commands

Table Creation

CREATE TABLE [*tableName*] **(**

[*fieldName*] [*type*] [NULL] [*options*],

PRIMARY KEY ([*fieldName*]) **);**

NULL : optionnal field

NOT NULL : required field

Options : AUTO_INCREMENT, DEFAULT,UNIQUE, CHECK

Table Commands

Modify the table structure

* **ALTER TABLE** [*tableName*] …

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

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

  DROP COLUMN [*fieldName*]**;**

* **DROP TABLE** [*tableName*]**;**

Data Commands

Insert datas

INSERT INTO [*tableName*] **(***fieldX*, …**)**

VALUES (***valueX*, …);**

Multiples insertions in one query is possible.

Data Commands

Insert datas

Muted columns :

- Key column with auto_increment > number automaticly inserted
- Column with value by default > default value inserted when no value is given
- Nullable column > Null is inserted when no value is given
- When no value is given to a column not nullable and with no default value,

  > the insertion is rejected

Data commands

Update existing datas

Update :

```
UPDATE tableName
SET fieldX = valueX
WHERE id = 1;
```

If no condition: the whole table is updated !

Data Commands

Delete datas

Delete :

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

If no condition: the whole table is impacted !

Empty a table:

```
TRUNCATE TABLE tableName ;
```

Data commands

Access datas

Read :

```
SELECT * FROM tableName;
```

Column alias:

```
SELECT price_df, price_df*1.20 AS price_it
FROM tableName ;
```

Data Commands

Filter datas

WHERE clause :

SELECT *fieldX, fieldY* **FROM** *tableName*

WHERE *condition1*

AND *condition2… ;*

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

Data Commands

Improve queries

ORDER BY clause :

Orders the datas (by text, by number, by dates)

LIMIT clause :

Limit the number of the results.

Data Commands

Data agregation

SUM	compute the sum of the column
AVG	compute the average of the column
MAX	give the maximum value of the column
MIN	give the minimum value of the column
COUNT	count the number of lines

Only in a SELECT

Data Commands

gathering

GROUP BY clause :

Gather the computed values of a column

```
SELECT id_genre, COUNT(id_book) AS nb_book
FROM book
GROUP BY id_genre ;
```

Data Commands

Keywords order of a SELECT

SELECT…

FROM…

    JOIN…

WHERE…

    AND/OR…

GROUP BY…

    HAVING…

ORDER BY…

LIMIT…

Advanced commands

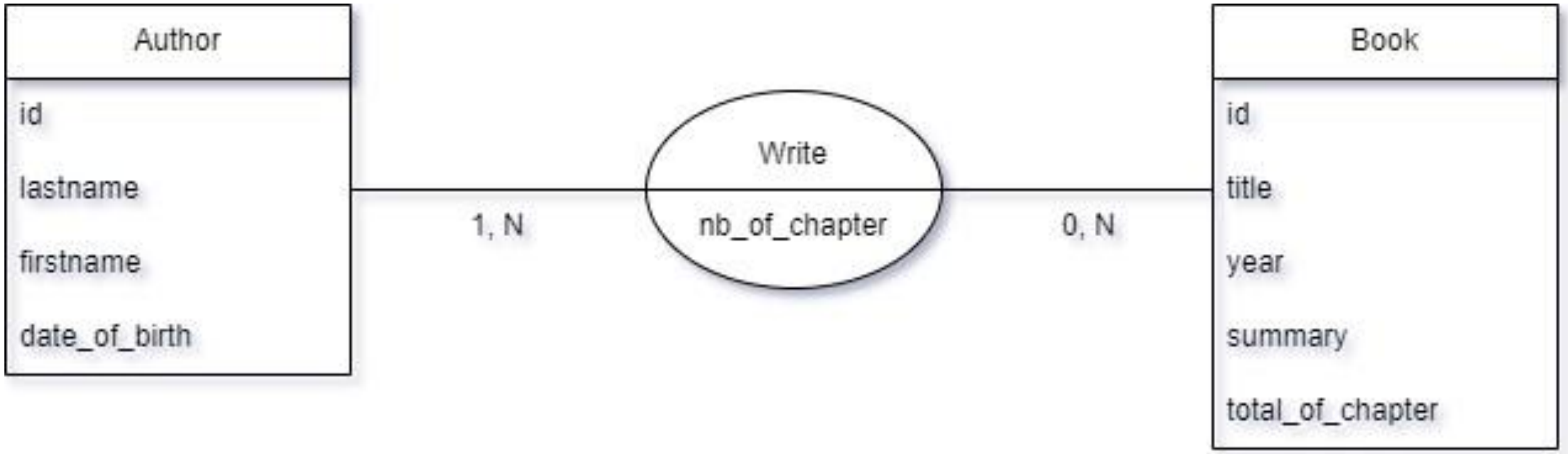
Joins : Multi-tables Queries

Merise Method

Conceptual Data Model (CDM)

Entities and Association:

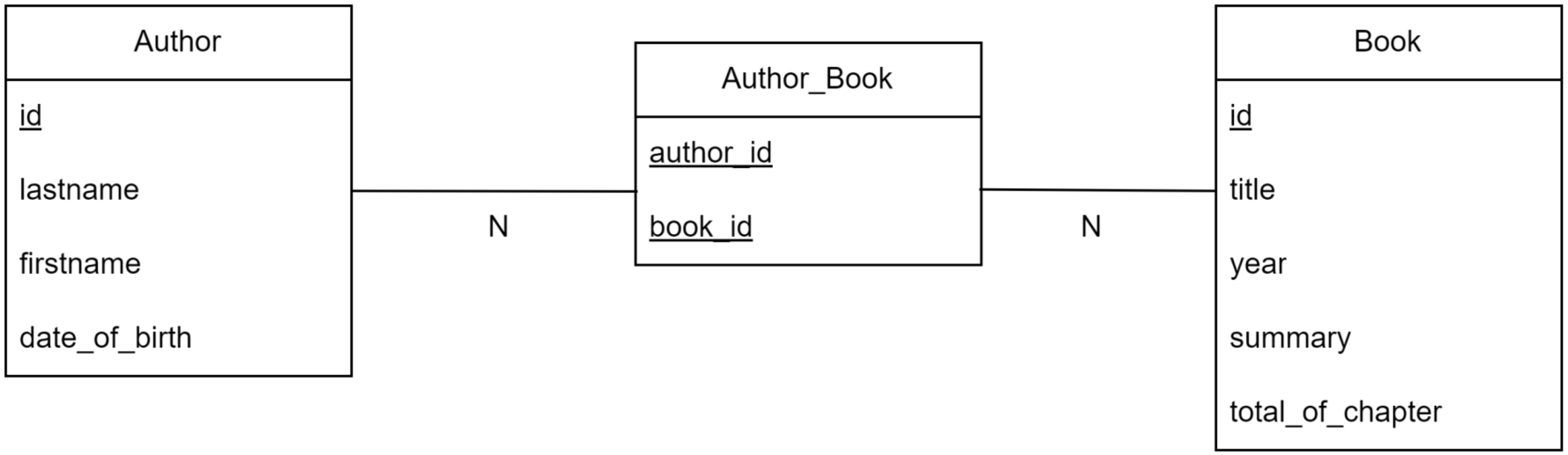
Observe how the cardinalities 0,N/1,N give informations on how the two table work together,



Merise Method

2. Logical Data Model (LDM)

- ✓ The LDM aims at describe <u>the data structure</u> without using any specifique technology.
- ✓ As such, this model is still indepedant to any DBMS.

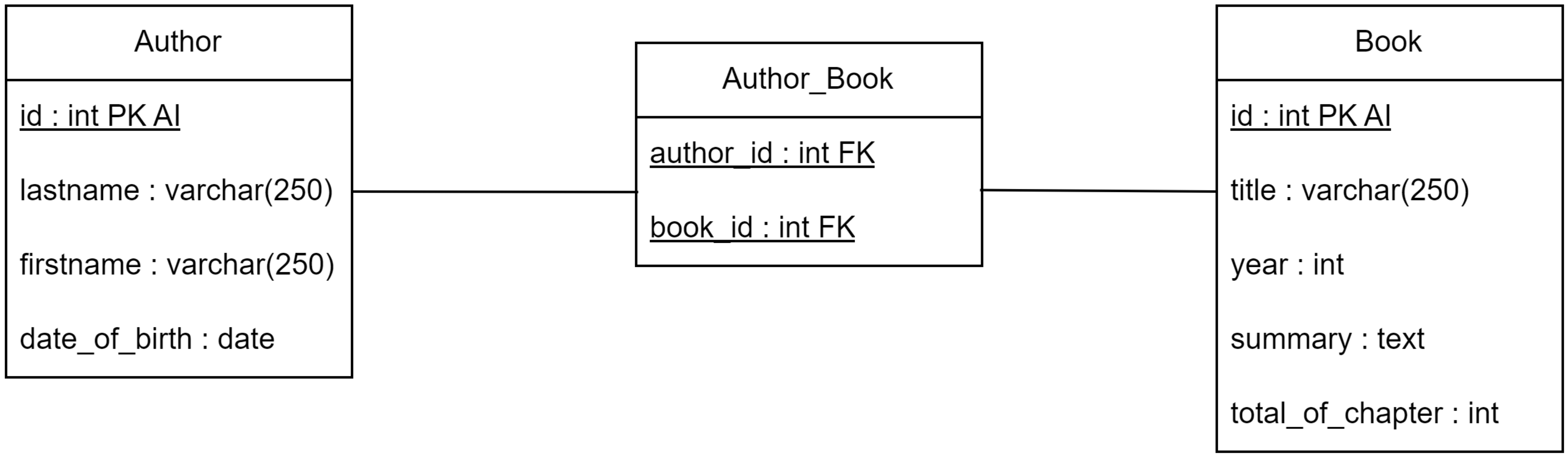


Merise Method

3. Physical Data Model (PDM)

✓ At this step, the MPD give the real type of the datas.

✓The MPD **IS** dependant of the final DBMS. (it will change if we change our DBMS)



Associations

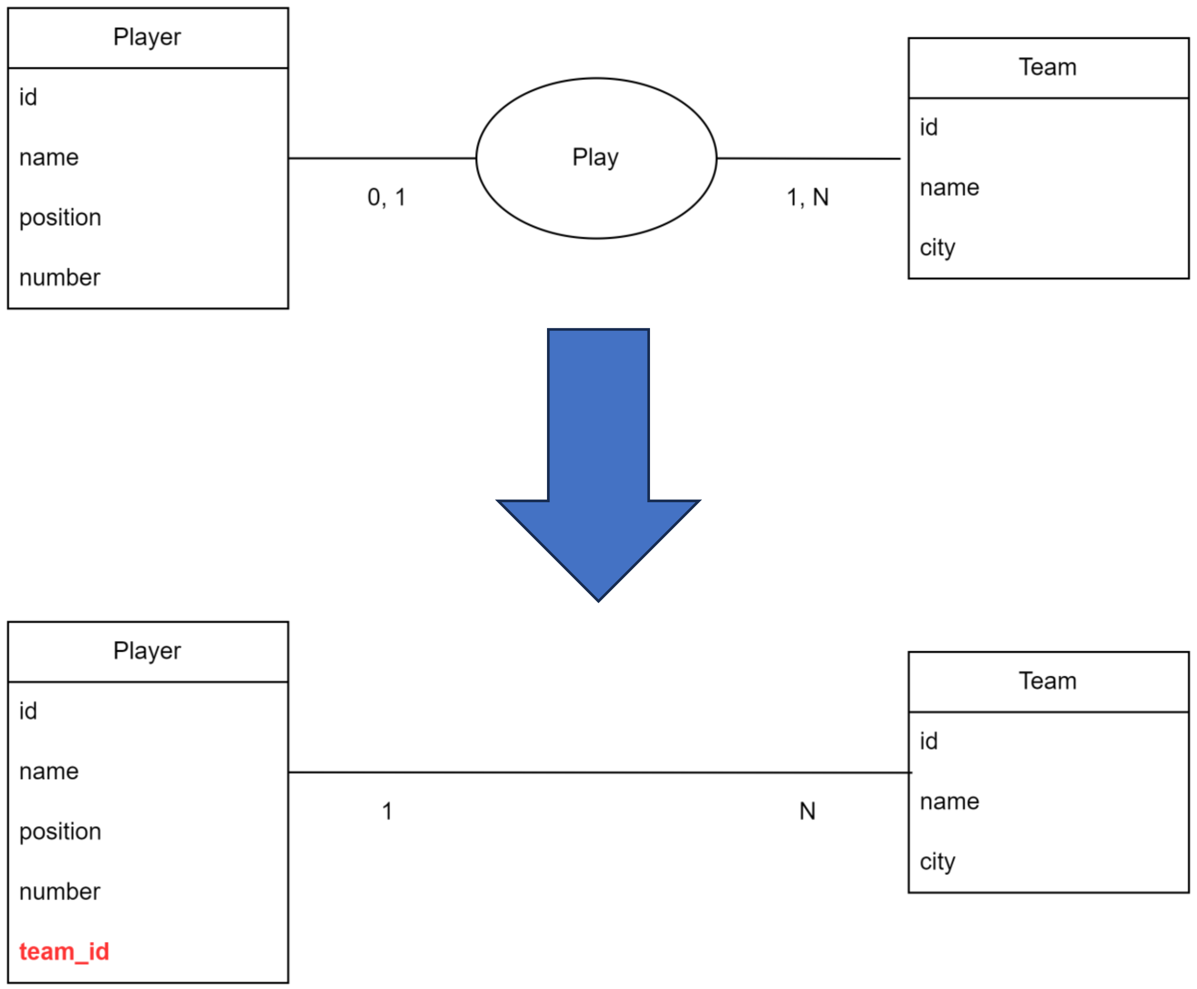
Between SQL tables – 3 types of associations are possibles

1. One – to – One
2. One – to – Many
3. Many – to – Many

One to many

MCD

MLD

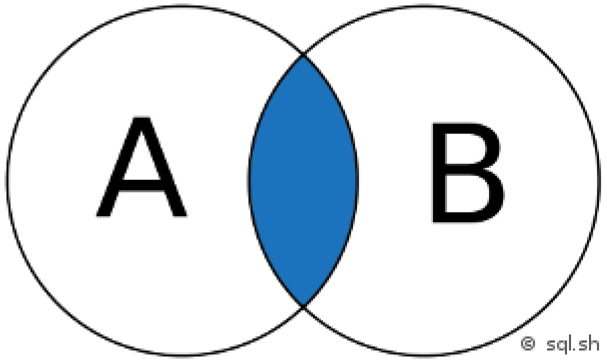


Advanced Select

Joins : Queries multi-tables

Joins :

- **INNER JOIN**



```
SELECT * FROM A
INNER JOIN B ON A.key = B.key
```

Advanced Select

Query INNER JOIN

Select datas on multiple tables

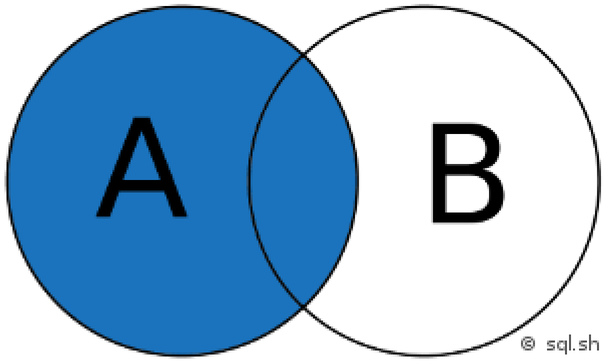
```
SELECT *
FROM tableA, tableB, tableC
WHERE tableA.id_tabB = tableB.id
AND tableA.id_tabC = tableC.id;
```

Advanced Select

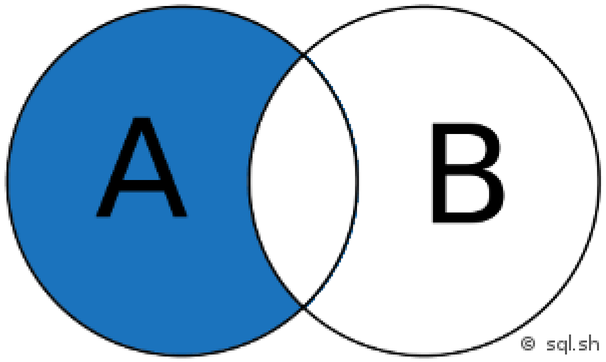
Joins : Queries multi-tables

- **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 ;
```

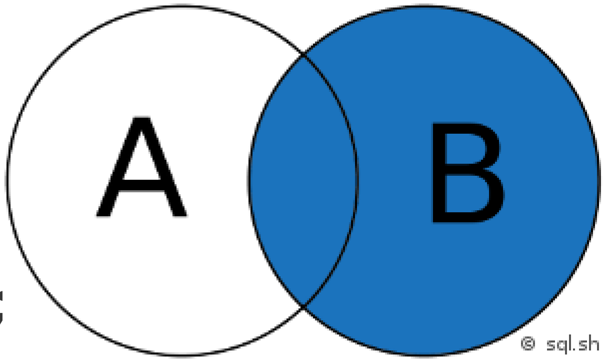


Advanced Select

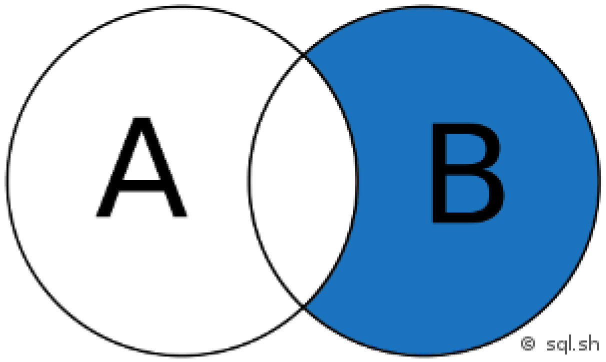
Joins : Queries multi-tables

- **RIGHT JOIN**

```
SELECT * FROM A
RIGHT JOIN B ON A.key = B.key ;
```



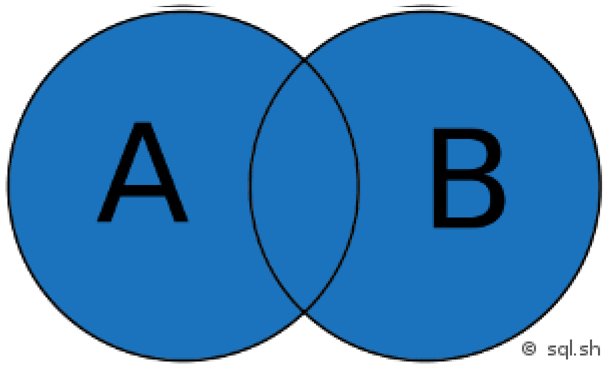
```
SELECT * FROM A
RIGHT JOIN B ON A.key = B.key
WHERE A.key IS NULL ;
```



Advanced Select

Joins : Queries multi-tables

- **FULL JOIN**



```
SELECT * FROM A
FULL JOIN B ON A.key = B.key
```

Introduction to RDBMS

Introduction to RDBMS

Relationnal Databases

- **Databases with relations :**
- *A table is linked to another (or more) by a relation, materialize throught a* ***foreign key****, following stricts constraints called « Integrity constraints.*

Introduction to RDBMS

Definition of a foreign key

Definition of keys :

/   MySQL Syntax */*

ALTER TABLE [*tableName1*]

ADD CONSTRAINT *ConstraintName* **(ex FK_table1_table2)**

FOREIGN KEY *tableName1* **(***id_table1***)**

REFERENCES [*tableName2*] **(***id_table2***) ;**

Integrity Constraint

Definition of chained actions:

… **ON UPDATE** [*value*] **ON DELETE** [*value*]

- ➢CASCADE :- *DROP TABLE nameTable CASCADE CONSTRAINTS;*
- ➢SET NULL
- ➢SET DEFAULT
- ➢RESTRICT

Advanced Usages

Advanced usages

Nested queries

Nest a query inside another to cross datas:

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

Advanced usages

The views : create / use

A view is virtual table based on SQL query

- ✓ Avoid to work with long queries : the view is a way to resume a query we use often.
- ✓ It allows to mask the real data model to some users.

Advanced usages

- **Create :**

CREATE VIEW *view_nomDeLaVue* **AS**

SELECT * FROM *nomDeLaVue* **;**

- **Modify :**

ALTER VIEW *nomDeLaVue* **AS**

SELECT …. ;

- **Delete :**

DROP VIEW *nomDeLaVue* **;**

Advanced usages

The views : limits

- **They can only be based on a SELECT**
- **One view = One query (Nesting is allowed)**

Advanced usages

The stocked procedures

- **Stocked procedure :**

```
DELIMITER //
CREATE PROCEDURE procName()
BEGIN
        ... SQL Queries
END//
```

Advanced usages

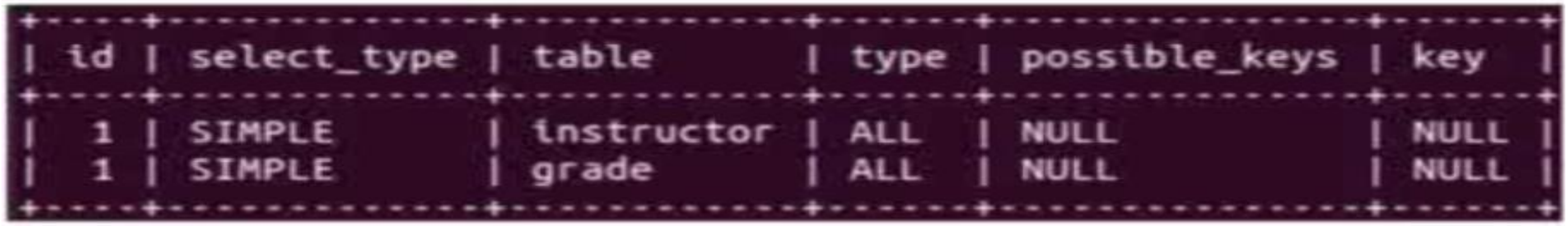
Optimisations

- **Check the weight of your queries**
- **Save your data ? Archive ?**

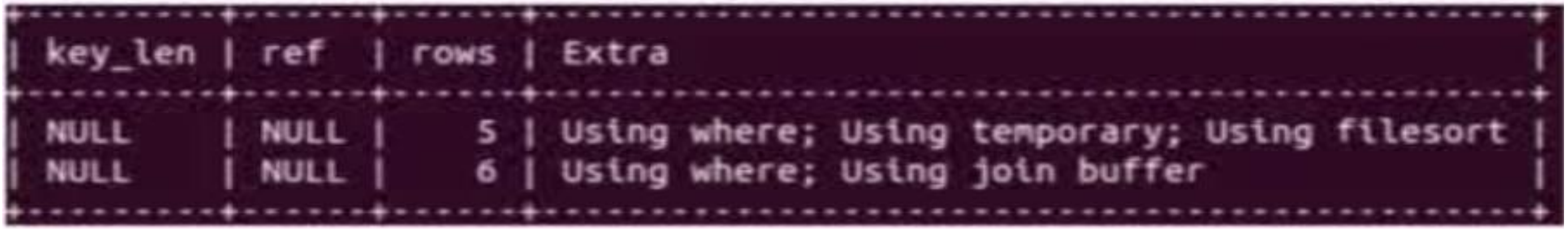
Optimisations

SQL Tool **Explain** :- *The EXPLAIN* statement provides information about how MySQL executes statements. ... EXPLAIN returns a row of information for each table used in the SELECT statement.

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 |
+----+-------------+------------+------+---------------+------+
```

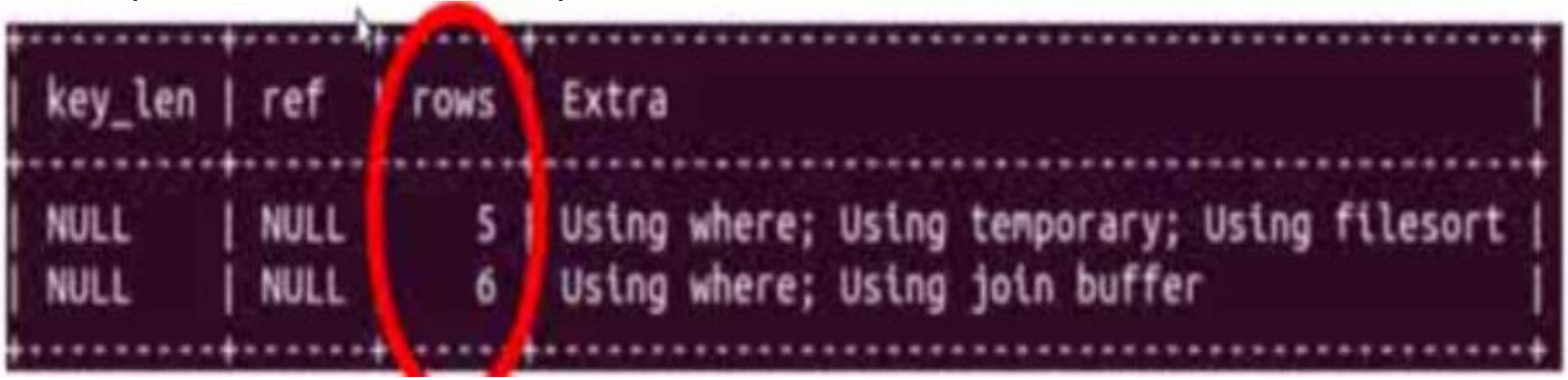


```
+---------+------+------+----------------------------------------------+
| key_len | ref  | rows | Extra                                        |
+---------+------+------+----------------------------------------------+
| NULL    | NULL |    5 | Using where; Using temporary; Using filesort |
| NULL    | NULL |    6 | Using where; Using join buffer               |
+---------+------+------+----------------------------------------------+
```

Advanced usages

Optimisations

- ✓Charge de la requête SQL
- ✓Interpréter les indicateurs Explain



```
+---------+------+------+---------------------------------------------------+
| key_len | ref  | rows | Extra                                             |
+---------+------+------+---------------------------------------------------+
| NULL    | NULL |    5 | Using where; Using temporary; Using filesort      |
| NULL    | NULL |    6 | Using where; Using join buffer                    |
+---------+------+------+---------------------------------------------------+
```

SQL