Objectif : Créer une DB où vs indiquer ds 1 table task ts sujets appris sur SQL (CREATE, ALTER, UPDATE...)

-- Pour suppr 1DB :

DROP DATABASE [IF EXISTS] database\_name;

DROP SCHEMA [IF EXISTS] database\_name;

-- Les 2 Qry sont équivalentes

-- Pour créer 1DB :

CREATE DATABASE [IF NOT EXISTS] database\_name

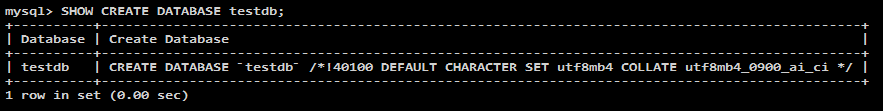
[CHARACTER SET charset\_name]

[COLLATE collation\_name]

-- Si le CHARCTER SET / COLLATE ne sont pas renseignés, MySQL créera la DB par défaut (latin1).

-- Pr retrouver le code de création, on lance SHOW :

SHOW CREATE DATABASE testdb; -- Par défaut : latin1



+----------+-------------------------------------------------------------------+

| Database | Create Database |

+----------+-------------------------------------------------------------------+

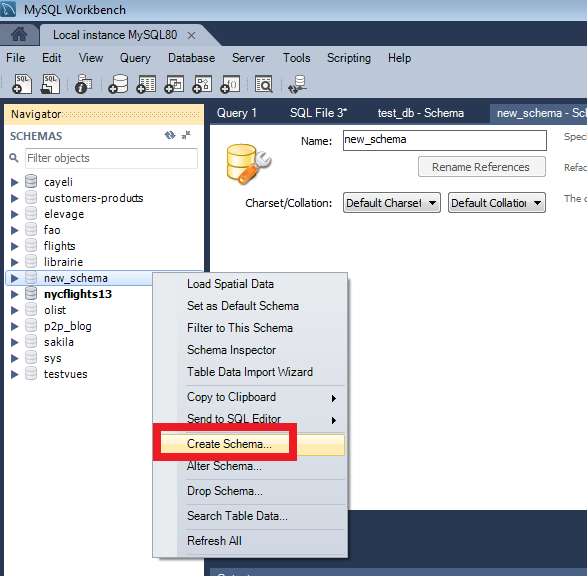
| testdb | CREATE DATABASE `testdb` /\*!40100 DEFAULT CHARACTER SET latin1 \*/ |

+----------+-------------------------------------------------------------------+

-- Q : Créer 2DB testdb1 (avec du code) et testdb2 (avec MySQLWorkBench ou HeidiSQL) et supp la testdb2 (avec du code).

CREATE DATABASE testdb /\* !40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ ;

-- Demo workbench/HeidiSQL pr la création de testdb2



-- Créer 1table

CREATE TABLE [IF NOT EXISTS] table\_name(

column\_name data\_type(length) [NOT NULL] [DEFAULT value] [AUTO\_INCREMENT],

PRIMARY KEY (col1,col2,...) -- Déclarer une PK

) ENGINE=storage\_engine ; -- Par défaut : InnoDB (gère les FK, transactions, ...)

-- Q : Créer 1table tasks dans la DB testdb ?

CREATE TABLE IF NOT EXISTS tasks (

task\_id INT ,

title VARCHAR(255) NOT NULL,

start\_date DATETIME,

end\_date DATETIME,

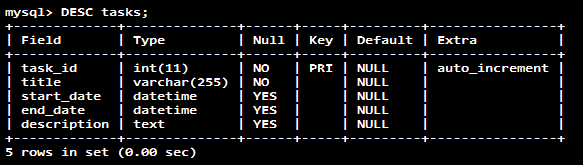
-- status TINYINT NOT NULL,

-- priority TINYINT NOT NULL,

description TEXT,

PRIMARY KEY (task\_id)

) ENGINE=INNODB;

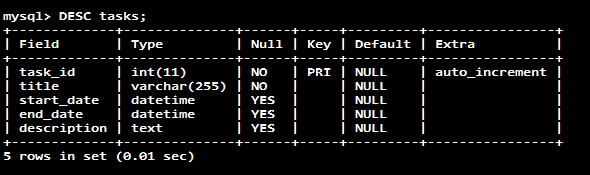


-- ALTER TABLE

-- Q : Modifier task\_id en AUTO\_INCREMENT ?

ALTER TABLE tasks

CHANGE COLUMN task\_id task\_id INT(11) AUTO\_INCREMENT;



-- Insérer des données pr le vérifier ?

INSERT INTO tasks(title, start\_date, end\_date, description)

VALUES('Learn MySQL CREATE TABLE',

Now(),

Now(),

'Practicing MySQL CREATE TABLE statement');

INSERT INTO tasks(title, start\_date, end\_date, description)

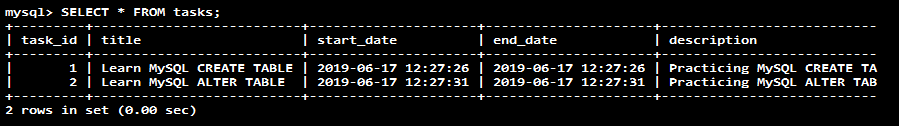
VALUES('Learn MySQL ALTER TABLE',

Now(),

Now(),

'Practicing MySQL ALTER TABLE statement');

SELECT \* FROM tasks ;



-- Cmnt fonctionne AUTO\_INCREMENT ?

-- Q : Créer 1table employees(emp\_no, first\_name, last\_name) ?

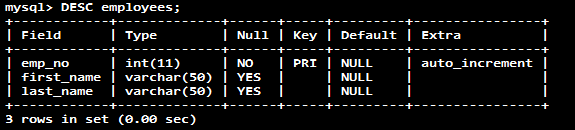
CREATE TABLE employees (

emp\_no INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50)

);



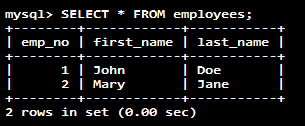
-- Q : Inserer les valeurs suivantes ?

INSERT INTO employees(first\_name,last\_name)

VALUES('John','Doe'),

('Mary','Jane');

SELECT \* FROM employees ;



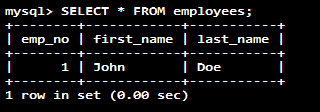
-- Q : Supp l'employé dont emp\_no = 2 ?

DELETE FROM employees

WHERE

emp\_no = 2;

SELECT \* FROM employees ;



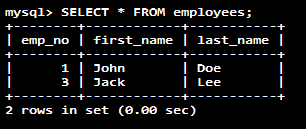
-- Q : Inserer un nouvel employé ?

INSERT INTO employees(first\_name,last\_name)

VALUES('Jack','Lee');

SELECT \* FROM employees ;

-- On passe à emp\_no = 3 (pas de recyclage des ID ; parc qu'on pense que MySQL lui passera emp\_no = 2)



-- Q : Faire en sorte que emp\_no = 3 prennent l'ID de emp\_no = 1

-- R : MySQL est garante de la fiabilité et la cohérence de la PK.

-- S : ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

UPDATE employees

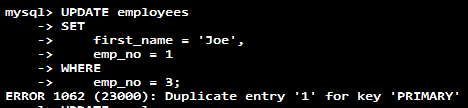
SET

first\_name = 'Joe',

emp\_no = 1

WHERE

emp\_no = 3;



-- Q : MAJ emp\_no = 3 à 10 ?

-- C'est possible de changer la valeur

UPDATE employees

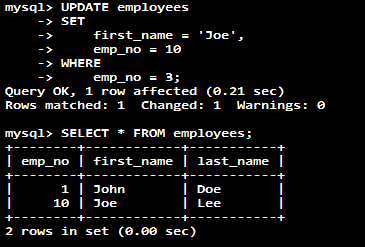
SET

first\_name = 'Joe',

emp\_no = 10

WHERE

emp\_no = 3;

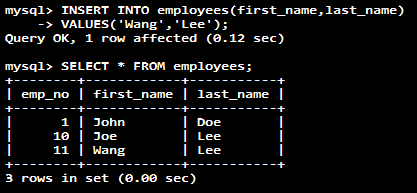


-- Q : Inserer un nouvel employe ?

-- R : On vérifie que MySQL garde le bon ordre des PK.

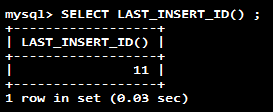
INSERT INTO employees(first\_name,last\_name)

VALUES('Wang','Lee');



-- Pr vérifier le dernier emp\_no renseigné :

SELECT LAST\_INSERT\_ID() ;



-- Gestion des NULL

-- Bonne pratique : Déclarer le plus svt cols en NOT NULL en affectant aussi une valeur par défaut.

-- La gestion des NULL rend les Qry plus complexes (donc plus d'effort de votre part).

Exemple :

CREATE TABLE inventory (

material\_no VARCHAR(18),

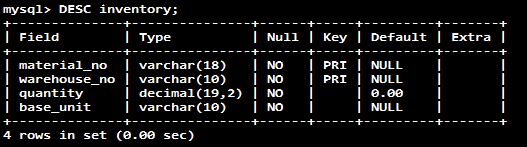
warehouse\_no VARCHAR(10),

quantity DECIMAL(19 , 2 ) NOT NULL DEFAULT 0,

base\_unit VARCHAR(10) NOT NULL,

PRIMARY KEY (material\_no , warehouse\_no)

);

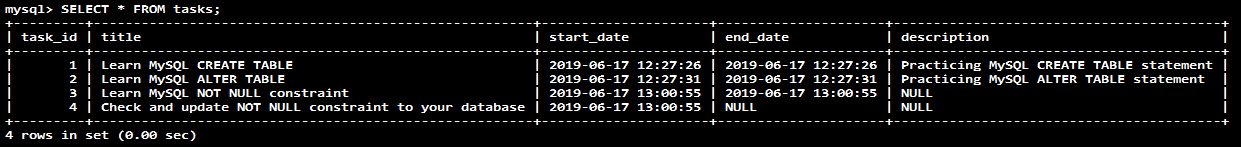


-- Q : Inserer ds la table 'tasks' ?

INSERT INTO tasks(title ,start\_date, end\_date)

VALUES('Learn MySQL NOT NULL constraint', Now(), Now()),

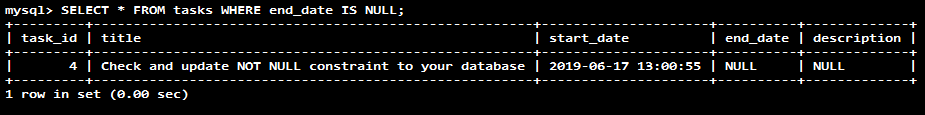
('Check and update NOT NULL constraint to your database', NOW(), NULL);



-- Ds l'exemple, la valeur par defaut pr la colonne 'quantity' est 0. Parce qu'au moment où vs ajouter une ligne ds la table 'inventory', la valeur de la quantity est 0, mais not NULL.

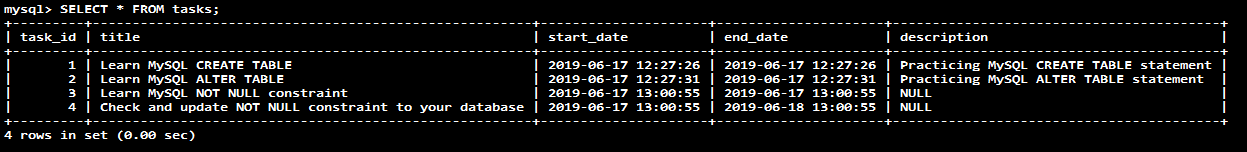
-- Q : Quelles sont les tasks dont la date de fin est manquantes ?

SELECT \* FROM tasks WHERE end\_date IS NULL;



-- Q : Remplir la date de fin (par ex + 24H) ?

UPDATE tasks SET end\_date = start\_date + INTERVAL 1 DAY WHERE end\_date IS NULL;

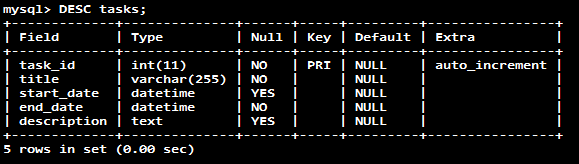


-- Q : Changer le type de la colonne "end\_date" en NOT NULL ?

ALTER TABLE tasks

CHANGE end\_date end\_date DATETIME NOT NULL;

DESC tasks ;



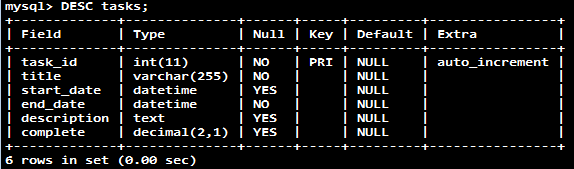
-- ALTER TABLE : ajouter/supp/renommer une colonne

-- Q : Ajouter une colonne "complete" apres "description" ?

ALTER TABLE tasks

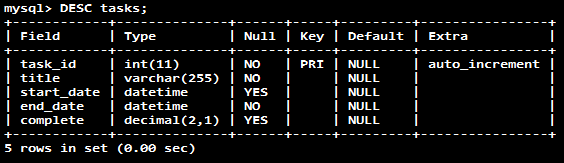
ADD COLUMN complete DECIMAL(2,1) NULL

AFTER description;



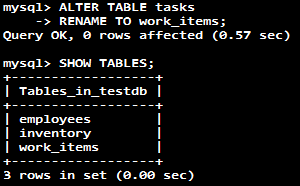
ALTER TABLE tasks

DROP COLUMN description;



ALTER TABLE tasks

RENAME TO work\_items;



-- Les PK, les FK en profondeur

CREATE TABLE users(

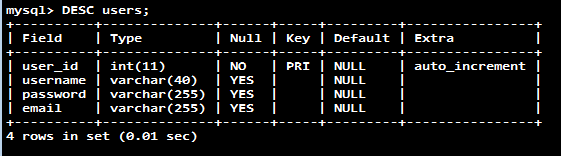
user\_id INT AUTO\_INCREMENT PRIMARY KEY, -- la PK est déclaré au début

username VARCHAR(40),

password VARCHAR(255),

email VARCHAR(255)

);



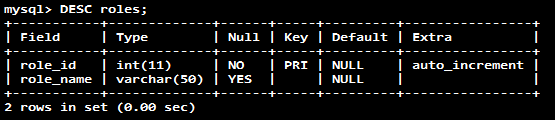
CREATE TABLE roles(

role\_id INT AUTO\_INCREMENT,

role\_name VARCHAR(50),

PRIMARY KEY(role\_id) -- la PK est déclaré à la fin

);



-- Table de jointure : relation many-to-many

CREATE TABLE user\_roles(

user\_id INT NOT NULL,

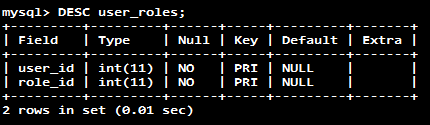
role\_id INT NOT NULL,

PRIMARY KEY(user\_id, role\_id), -- PK composite

FOREIGN KEY(user\_id) REFERENCES users(user\_id),

FOREIGN KEY(role\_id) REFERENCES roles(role\_id)

);



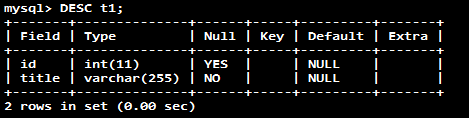
-- définir la PK avec ALTER TABLE

CREATE TABLE t1(

id INT,

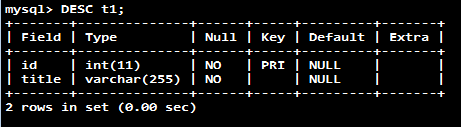
title VARCHAR(255) NOT NULL

);



ALTER TABLE t1

ADD PRIMARY KEY(id);



-- FK en profondeur

-- sans ALTER TABLE

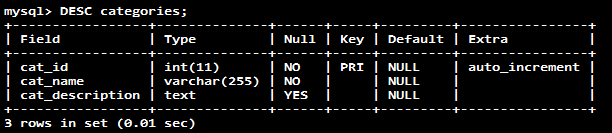
CREATE TABLE categories(

cat\_id int not null auto\_increment primary key,

cat\_name varchar(255) not null,

cat\_description text

) ENGINE=InnoDB;



CREATE TABLE products(

prd\_id int not null auto\_increment primary key,

prd\_name varchar(355) not null,

prd\_price decimal,

cat\_id int not null,

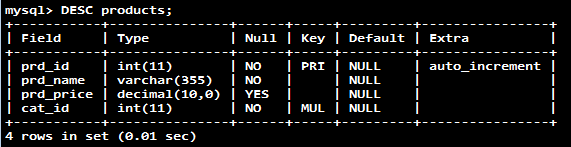
FOREIGN KEY fk\_cat(cat\_id)

REFERENCES categories(cat\_id)

ON UPDATE CASCADE

ON DELETE RESTRICT -- Rejet de tte suppression

)ENGINE=InnoDB;



-- Avec ALTER TABLE

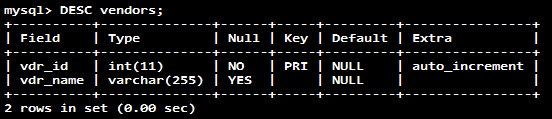
-- Q : Créer une nouvelle table vendors ?

CREATE TABLE vendors(

vdr\_id int not null auto\_increment primary key,

vdr\_name varchar(255)

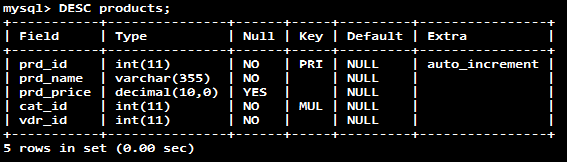
)ENGINE=InnoDB;



-- Q : Ajouter à la table products le vendor\_id

ALTER TABLE products

ADD COLUMN vdr\_id int not null AFTER cat\_id;



-- Q : Déclarer la FK ?

ALTER TABLE products

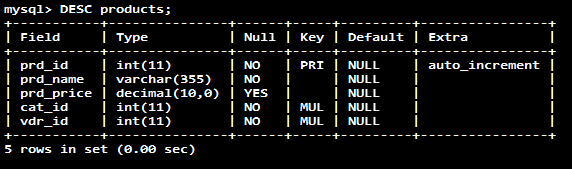
ADD FOREIGN KEY fk\_vendor(vdr\_id)

REFERENCES vendors(vdr\_id)

ON DELETE NO ACTION

ON UPDATE CASCADE;

-- Maintenant, la table "products" possède 2 FK, la 1è refere vers "categories" table et la 2è refere vers la table "vendors".

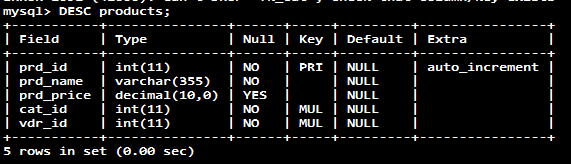


-- Q : Supp une FK ?

ALTER TABLE products

DROP FOREIGN KEY fk\_cat;

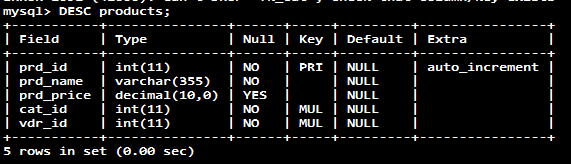
NOTE : on voit que la variable cat\_id est toujours en MUL (FK).



ALTER TABLE products

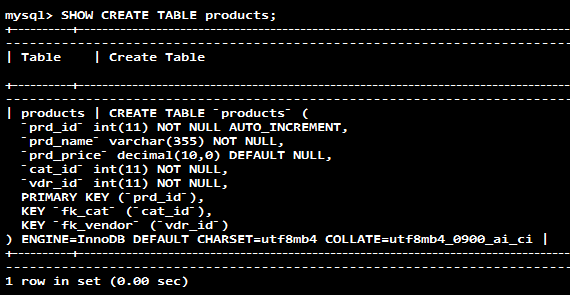
DROP FOREIGN KEY products\_ibfk\_2;

NOTE : on voit que la variable vdr\_id est toujours en MUL (FK).



🡪 Pour voir que les FK ont bien été supprimées, lancer la commande :

SHOW CREATE TABLE products;



-- Q : Désactiver la FK ?

SET foreign\_key\_checks = 0;

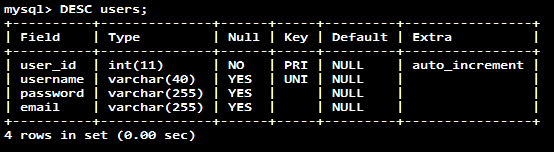


-- Ajouter une contrainte d'Unicité

-- On souhaite ajouter une contrainte : username et email soient uniques

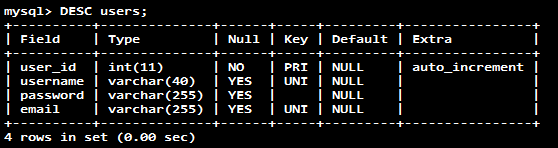
ALTER TABLE users

ADD UNIQUE INDEX username\_unique (username ASC) ;



ALTER TABLE users

ADD UNIQUE INDEX email\_unique (email ASC) ;



-- Autre exemple d'unicité

CREATE TABLE IF NOT EXISTS suppliers (

supplier\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

phone VARCHAR(12) NOT NULL UNIQUE,

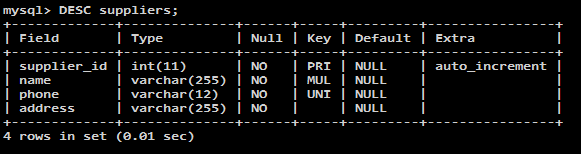
address VARCHAR(255) NOT NULL,

CONSTRAINT uc\_name\_address UNIQUE (name , address)

);

-- S : Chq suppliers a un num de tél unique !

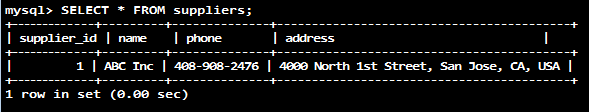
-- S : 2 suppliers ne peuvent pas avoir le même nom et la même à la fois !



-- 1è insertion pr tester les contraintes

INSERT INTO suppliers(name, phone, address)

VALUES('ABC Inc', '408-908-2476','4000 North 1st Street, San Jose, CA, USA');



-- 2è insertion avec le même num de tél

INSERT INTO suppliers(name, phone, address)

VALUES('XYZ Corporation', '408-908-2476','4001 North 1st Street, San Jose, CA, USA');

error :

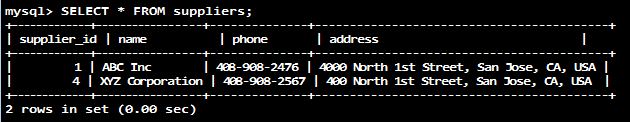
Error Code: 1062. Duplicate entry '408-908-2476' for key 'phone'



-- Corrigeons le Pb :

INSERT INTO suppliers(name, phone, address)

VALUES('XYZ Corporation', '408-908-2567','400 North 1st Street, San Jose, CA, USA');



-- Test sur la 2è contrainte : nom et adresse (unique)

INSERT INTO suppliers(name, phone, address)

VALUES('XYZ Corporation', '408-908-102','400 North 1st Street, San Jose, CA, USA');

-- Violation de la contrainte d'unicité

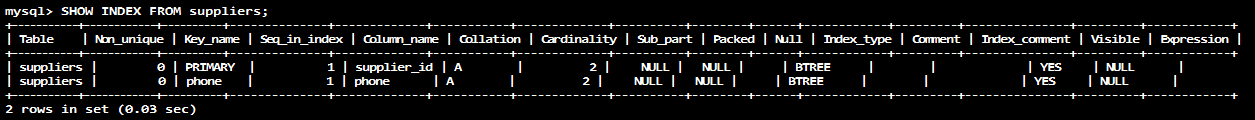
Error Code: 1062. Duplicate entry 'XYZ Corporation-400 North 1st Street, San Jose, CA, USA' for key 'name'



Q : Supp un INDEX ?

DROP INDEX uc\_name\_address ON suppliers;

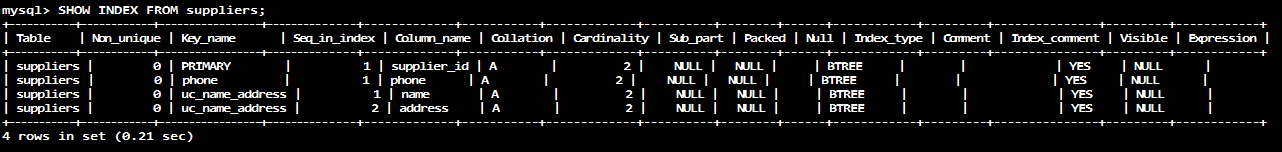
SHOW INDEX FROM suppliers;



-- Instaurer une conrainte avec ALTER TABLE

ALTER TABLE suppliers

ADD CONSTRAINT uc\_name\_address UNIQUE (name,address);



~~-- Cmnt valider vos données (CHECK) ?~~

~~CREATE TABLE IF NOT EXISTS parts (~~

~~part\_no VARCHAR(18) PRIMARY KEY,~~

~~description VARCHAR(40),~~

~~cost DECIMAL(10 , 2 ) NOT NULL CHECK(cost > 0), -- cost doit être positif~~

~~price DECIMAL (10,2) NOT NULL~~

~~);~~

CREATE TABLE IF NOT EXISTS parts (

part\_no VARCHAR(18) PRIMARY KEY,

description VARCHAR(40),

cost DECIMAL(10 , 2 ) NOT NULL CHECK (cost > 0),

price DECIMAL(10 , 2 ) NOT NULL CHECK (price > 0),

CHECK (price >= cost) -- prix doit être sup ou égal au coût

);

