Advanced Database

Lab 1

Creating a Database

Tutor : M. SOLTANI ROOZBEH

Student : ALEX DONG

Class : MSc 1 DMIA (inter)

Table of contents

Objective

In this lab, we design and develop the database of some school, with a focus on students, courses and enrolments.

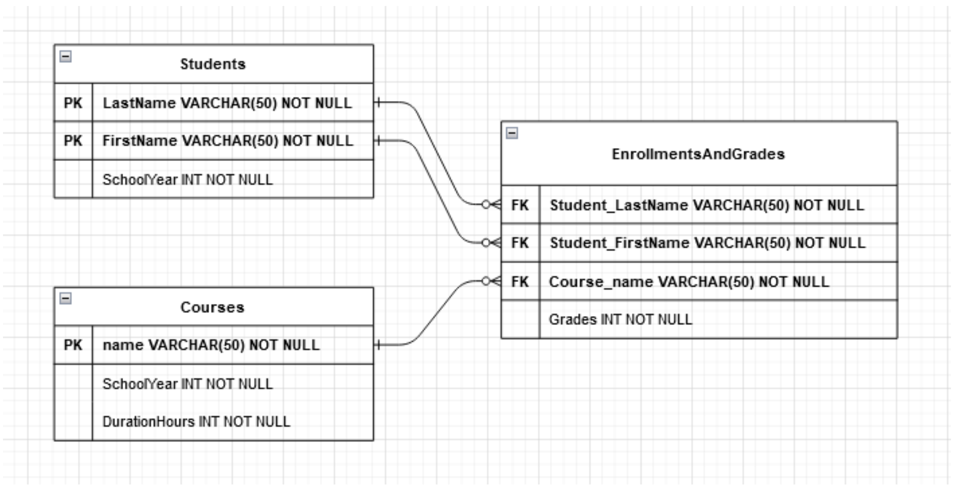
1 Database Modeling

The schema of the relation model of database will be the following, by using the conventional notation to represent the tables, primary keys (PK) and foreign keys (FK) :

Table "Students"
PK : (LastName, FirstName)
Attributes : LastName, FirstName, SchoolYear
The primary key is the attributes LastName and FirstName. This means that each combination of LastName and FirstName must be unique, ensuring that each student is uniquely identified by their name.

Table "Courses"
PK : Name
Attributes : Name, SchoolYear, durationHours
The primary key is the attribute Name. This means that each course must have a unique name.

Table "EnrollmentsAndGrades"
FK : (Student_LastName, Student_FirstName) , references of the table « Students »
FK : Course_Name , references of the table « Courses »
Attributes : Student_LastName, Student_FirstName, Course_Name, Grades
This table represents the enrollment of students in courses and their corresponding grades. It uses foreign keys (FKs) to link to the "Students" and "Courses" tables.


Students
PK LastName VARCHAR(50) NOT NULL
PK FirstName VARCHAR(50) NOT NULL
SchoolYear INT NOT NULL
EnrollmentsAndGrades
FK Student_LastName VARCHAR(50) NOT NULL
FK Student_FirstName VARCHAR(50) NOT NULL
FK Course_name VARCHAR(50) NOT NULL
Grades INT NOT NULL
Courses
PK name VARCHAR(50) NOT NULL
SchoolYear INT NOT NULL
DurationHours INT NOT NULL

This schema allows students to be enrolled in multiple courses, because the table "EnrollmentAndGrades" use a primary key composed of Student_LastName, Student_FirstName and

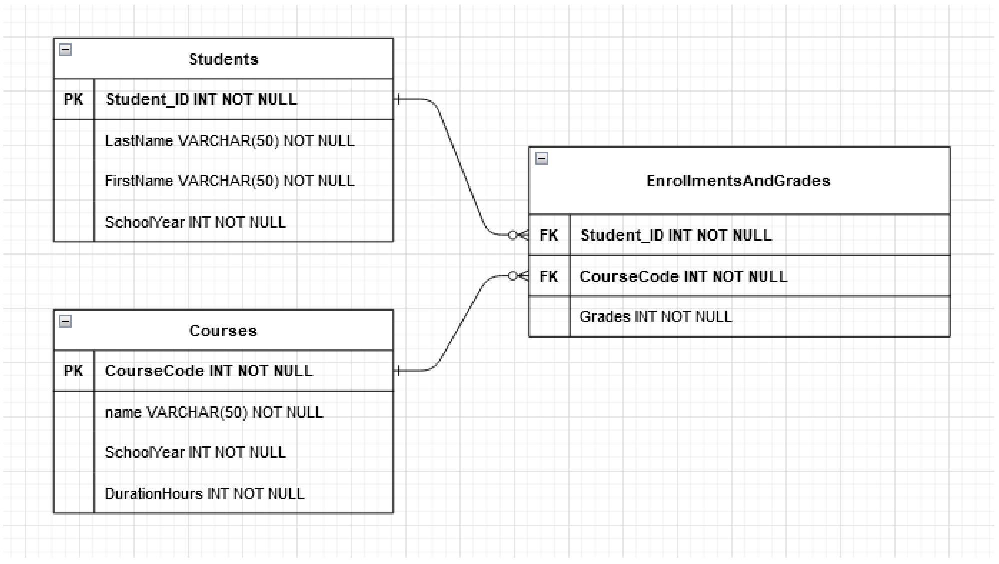
CoursesCode, to identify uniquely each enrollment. A student can finally have multiple records in the table “EnrollmentsAndGrades”.

Moreover, this schema allows also to a course not having a student registered because the « EnrollmentAndGrades » table can contain courses without any correspondence in the “Students” table.

2 Database Modeling – Alternate Solution

We now assume that students also have a student ID, and that courses also have a unique code, e.g. INGPA-INF4000-13 for the database course.

Considering the new assumptions that students have a unique student ID and that course have unique code, here is the new relational schema of the DB :



With this new schema, each student is identified by their unique student ID, and each course is identified by its unique code. The table « EnrollmentAndGrades » is linked to both the student and the course via corresponding foreign key.

This schema still allows a student to be enrolled in multiple courses and a course can have multiple students enrolled. Moreover, it considers the possibility that multiple courses may have the same name but different course codes, because the course code is now a unique primary key to identify courses.

3 Comparing the Two Models

Compare the relational schemas of exercises 1 and 2 with respect to:

- **disk space usage :**

  The schema of exercise 1 may result in high disk space usage, due of duplication of the student LastName and FirstName information. The schema of exercise 2 is more efficient in terms of disk space usage because it avoids information redundancy.

- **the query "find the name of the courses Salim Dupond is enrolled in":**

  In the exercise 1, if we execute a query to find "Salim Dupond", we would search for all table records that correspond to his FirstName and LastName. This requires searching the whole table, which may be less efficient. So, using an ID as primary key can be more efficient in term of performance because the search will be done on unique key.

- **the modification "change the course name databases for advanced databases":**

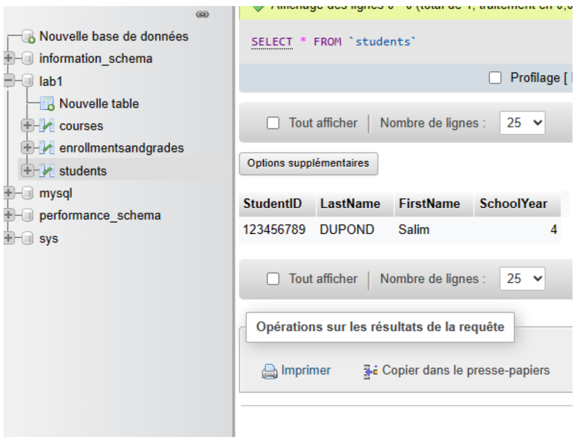
  If we want to modify the course name in the database of schema 1, we would update all records of the table " Courses". This operation can be costly in term of resources. So, it would be easier to update the course name using the unique course code as a reference. This approach allows more efficient.

4 Creating and Populating the Database

```sql
-- Create the Students table
CREATE TABLE Students (
     Student_ID VARCHAR(10),
     LastName VARCHAR(50),
     FirstName VARCHAR(50),
     SchoolYear INT
);

-- Create the Courses table
CREATE TABLE Courses (
     CourseCode VARCHAR(20),
     Name VARCHAR(50),
     SchoolYear INT,
     durationHours INT
);

-- Create the EnrollmentsAndGrades table
CREATE TABLE EnrollmentsAndGrades (
     Student_ID VARCHAR(10),
     CourseCode VARCHAR(20),
     Grades INT
);
```

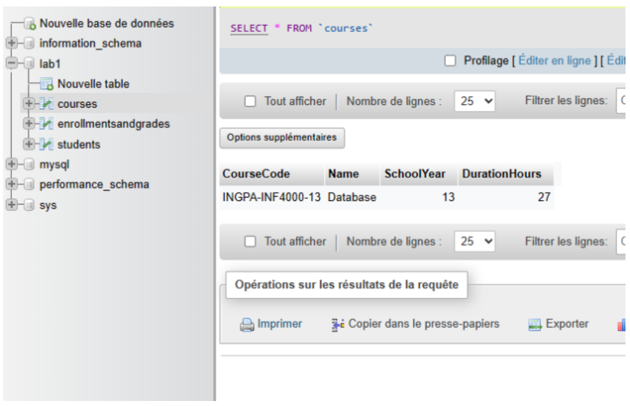
```sql
-- Populate the Students table
INSERT INTO Students (Student_ID, LastName, FirstName, SchoolYear)
VALUES ('123456789', 'DUPONT', 'Salim', 4);

-- Populate the Courses table
INSERT INTO Courses (CourseCode, Name, SchoolYear, durationHours)
VALUES ('INGPA-INF4000-13', 'Databases', 13, 27);

-- Populate the EnrollmentsAndGrades table
INSERT INTO EnrollmentsAndGrades (Student_ID, CourseCode, Grades)
VALUES ('123456789', 'INGPA-INF4000-13', 18);
```

We have now created and populated our database without primary and foreign key constraints.



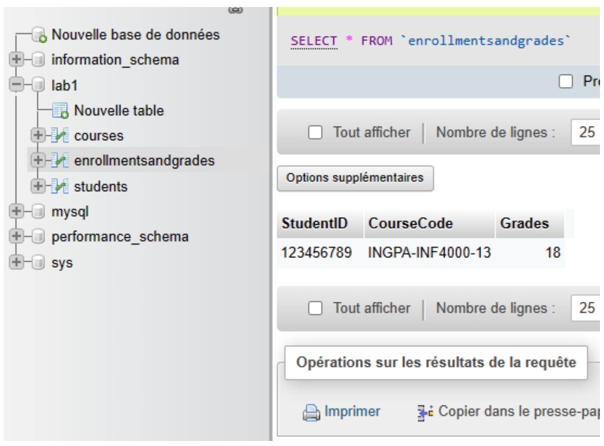
```sql
SELECT * FROM `students`
```

StudentID	LastName	FirstName	SchoolYear
123456789	DUPOND	Salim	4



```sql
SELECT * FROM `courses`
```

CourseCode	Name	SchoolYear	DurationHours
INGPA-INF4000-13	Database	13	27



```sql
SELECT * FROM `enrollmentsandgrades`
```

StudentID	CourseCode	Grades
123456789	INGPA-INF4000-13	18

Regarding the observation about modifying and querying the database without primary or foreign keys, it's important to note that primary and foreign keys are constraints that help maintain data integrity and enforce referential integrity in a database.

Even without these constraints, we can still perform some SQL operations, but we can risk having incorrect data. So, adding constraints can help prevent data anomalies and errors.

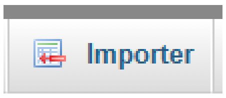
5 Database Script

To create a database script named “school.sql” with all the SQL statements required to create, populate, and define primary and foreign key constraints, here’s how to do :

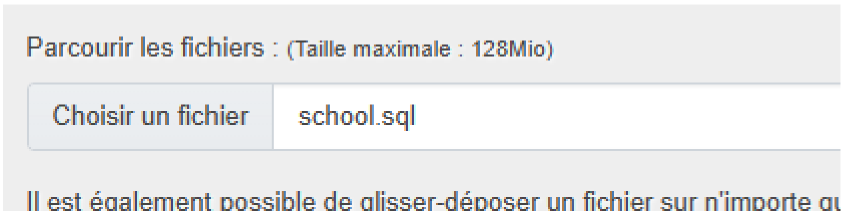
“I wrote this script on VS CODE”.

```sql
-- Drop tables if they exist
DROP TABLE IF EXISTS EnrollmentsAndGrades;
DROP TABLE IF EXISTS Courses;
DROP TABLE IF EXISTS Students;

-- Create the Students table
CREATE TABLE Students (
     Student_ID VARCHAR(10) PRIMARY KEY,
     LastName VARCHAR(50),
     FirstName VARCHAR(50),
     SchoolYear INT,
);

-- Create the Cours table
CREATE TABLE Courses (
     CourseCode VARCHAR(20) PRIMARY KEY,
     Name VARCHAR(255),
     SchoolYear INT,
     durationHours INT
);

-- Create the EnrollmentsAndGrades table with foreign key constraints
CREATE TABLE EnrollmentsAndGrades (
     Student_ID VARCHAR(10),
     CourseCode VARCHAR(20),
     Grades INT,
     CONSTRAINT pk_enrollmentsandgrades PRIMARY KEY (Student_ID, CourseCode),
     CONSTRAINT fk_student_enroll FOREIGN KEY (Student_ID) REFERENCES Students
(Student_ID),
     CONSTRAINT fk_course_enroll FOREIGN KEY (CourseCode) REFERENCES Courses
(CourseCode)
);

-- Populate the Students table
INSERT INTO Students (Student_ID, LastName, FirstName, SchoolYear)
VALUES ('123456789', 'DUPONT', 'Salim', 4);
-- Populate the Courses table
INSERT INTO Courses (CourseCode, Name, SchoolYear, durationHours)
VALUES ('INGPA-INF4000-13', 'Databases', 13, 27);

-- Populate the EnrollmentAndGrades table
INSERT INTO EnrollmentAndGrades (Student_ID, CourseCode, Grades)
VALUES ('123456789', 'INGPA-INF4000-13', 18);
```

Click on the import button, to import the script “school.sql”



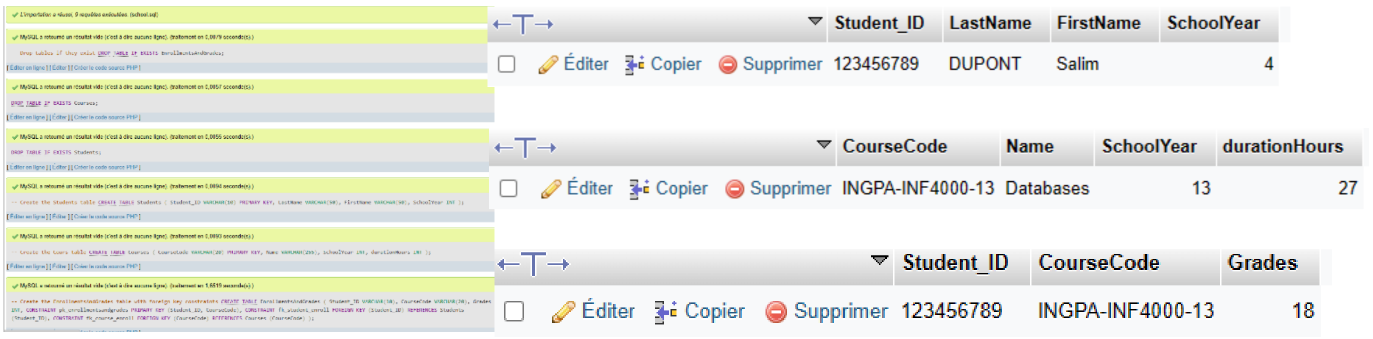
After that, choose the file.


Parcourir les fichiers : (Taille maximale : 128Mio)

Choisir un fichier school.sql

Il est également possible de glisser-déposer un fichier sur n'importe qu

And click on the button “import.”



```
✓ L'importation a réussi, 9 requêtes exécutées. (school.sql)

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0079 seconde(s).)
  Drop tables if they exist DROP TABLE IF EXISTS EnrollmentsAndGrades;
[ Éditer en ligne ] [ Éditer ] [ Créer le code source PHP ]

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0057 seconde(s).)
DROP TABLE IF EXISTS Courses;
[ Éditer en ligne ] [ Éditer ] [ Créer le code source PHP ]

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0055 seconde(s).)
DROP TABLE IF EXISTS Students;
[ Éditer en ligne ] [ Éditer ] [ Créer le code source PHP ]

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0094 seconde(s).)
-- Create the Students table CREATE TABLE Students ( Student_ID VARCHAR(10) PRIMARY KEY, LastName VARCHAR(50), FirstName VARCHAR(50), SchoolYear INT );
[ Éditer en ligne ] [ Éditer ] [ Créer le code source PHP ]

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0093 seconde(s).)
-- Create the Cours table CREATE TABLE Courses ( CourseCode VARCHAR(20) PRIMARY KEY, Name VARCHAR(255), SchoolYear INT, durationHours INT );
[ Éditer en ligne ] [ Éditer ] [ Créer le code source PHP ]

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 1,6519 seconde(s).)
-- Create the EnrollmentsAndGrades table with foreign key constraints CREATE TABLE EnrollmentsAndGrades ( Student_ID VARCHAR(10), CourseCode VARCHAR(20), Grades INT, CONSTRAINT pk_enrollmentsandgrades PRIMARY KEY (Student_ID, CourseCode), CONSTRAINT fk_student_enroll FOREIGN KEY (Student_ID) REFERENCES Students (Student_ID), CONSTRAINT fk_course_enroll FOREIGN KEY (CourseCode) REFERENCES Courses (CourseCode) );
```

←T→	Student_ID	LastName	FirstName	SchoolYear
☐ Éditer Copier Supprimer	123456789	DUPONT	Salim	4

←T→	CourseCode	Name	SchoolYear	durationHours
☐ Éditer Copier Supprimer	INGPA-INF4000-13	Databases	13	27

←T→	Student_ID	CourseCode	Grades
☐ Éditer Copier Supprimer	123456789	INGPA-INF4000-13	18

In this script, we start by dropping the tables if they already exist to ensure a fresh start. We create the tables in the following order: Students, Courses, and EnrollmentsAndGrades. This order respects the foreign key constraints.

Please note that the order of table creation and dropping is important to avoid foreign key constraint violations.

Running this script multiple times can correctly create and populate the database from scratch each time.

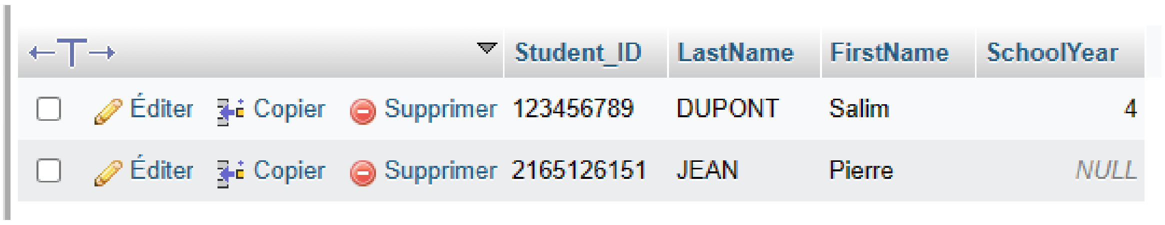
6 Modifying the Database’s Instance

Using an SQL interpreter:

- Insert a new student whose school year is not known (2 methods).

Method 1 (allow NULL)

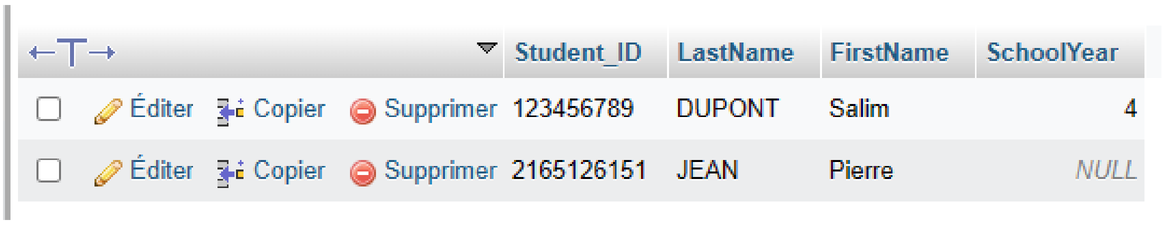
```
INSERT INTO Students (Student_ID, LastName, FirstName)
VALUES ('2165126151', 'JEAN', 'Pierre');
```



←T→	Student_ID	LastName	FirstName	SchoolYear
☐ Éditer Copier Supprimer	123456789	DUPONT	Salim	4
☐ Éditer Copier Supprimer	2165126151	JEAN	Pierre	NULL

Method2 (setting NULL):

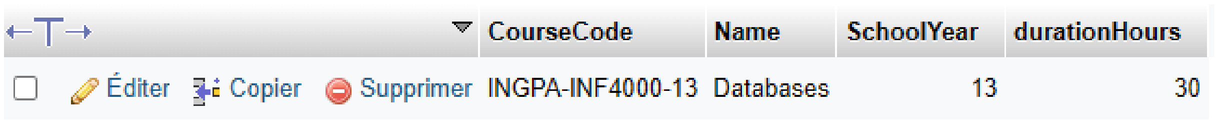
```
INSERT INTO Students (Student_ID, LastName, FirstName, SchoolYear)
VALUES ('2165126151', 'JEAN', 'Pierre', NULL);
```



				Student_ID	LastName	FirstName	SchoolYear
☐	Éditer	Copier	Supprimer	123456789	DUPONT	Salim	4
☐	Éditer	Copier	Supprimer	2165126151	JEAN	Pierre	NULL

- Change the duration of the Database course from 27 to 30 hours.

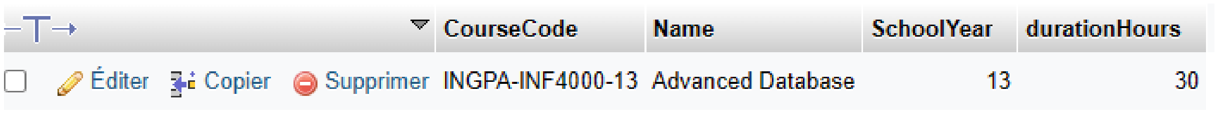
```
UPDATE Courses
SET durationHours = 30
WHERE CourseCode = 'INGPA-INF4000-13';
```



				CourseCode	Name	SchoolYear	durationHours
☐	Éditer	Copier	Supprimer	INGPA-INF4000-13	Databases	13	30

- Change the name of the Databases course to Advanced Databases

```
UPDATE Courses
SET Name = 'Advanced Databases'
WHERE CourseCode = 'INGPA-INF4000-13';
```



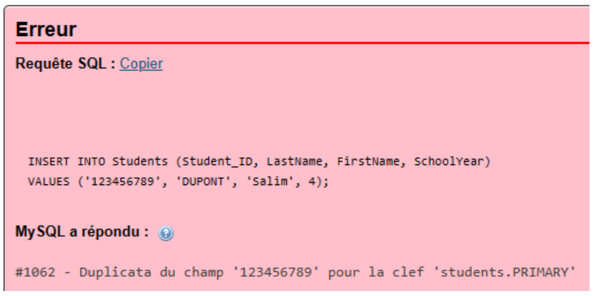
				CourseCode	Name	SchoolYear	durationHours
☐	Éditer	Copier	Supprimer	INGPA-INF4000-13	Advanced Database	13	30

7 Checking the Key Constraints

Primary key constraint on table Students:

1. Try to insert the same student twice. Next, try to insert a new student with a NULL ID. Observe the error message that you get and explain it.

If we insert the same student, we will get a display error, in which the ID of student who is a primary key, cannot be duplicated, and ensure the uniquely of the student.



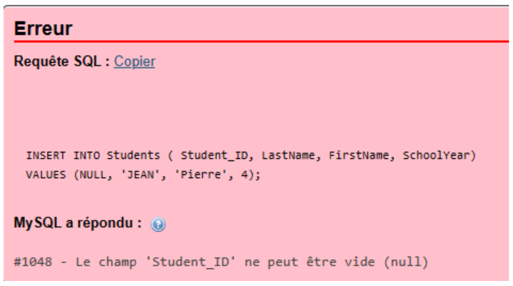
Erreur

Requête SQL : Copier

```
INSERT INTO Students (Student_ID, LastName, FirstName, SchoolYear)
VALUES ('123456789', 'DUPONT', 'Salim', 4);
```

MySQL a répondu :

```
#1062 - Duplicata du champ '123456789' pour la clef 'students.PRIMARY'
```



Erreur

Requête SQL : Copier

```
INSERT INTO Students ( Student_ID, LastName, FirstName, SchoolYear)
VALUES (NULL, 'JEAN', 'Pierre', 4);
```

MySQL a répondu :

```
#1048 - Le champ 'Student_ID' ne peut être vide (null)
```

If we insert a new student with a NULL ID into a table with primary key constraint we will encounter a violation of the primary key constraint, a primary key cannot be NULL.

2. Drop the primary key constraint on table Students and perform again the two inserts of the previous questions. What do you observe now?

After dropping the primary key constraint, the database will allow you to insert the same student multiple times and also insert students with NULL IDs without any error. The database will no longer enforce uniqueness on the "StudentID" column, and NULL values will be accepted.

3. Try to restore the primary key constraint on table Students. Explain what you observe.

When we try to restore the primary key constraint on the "StudentID" column, the database enforce uniqueness again. If there are duplicate or NULL values in the "StudentID" column, we got an error. We must resolve any duplicates or NULL values before reapplying the primary key constraint.

4. Modify the instance of the table so that you can now restore the primary key constraint.

To restore the primary key constraint, you need to ensure that the "StudentID" column has unique values for each student, and there are no NULL values.

We may need to update or delete rows to achieve this. Once the data meets the uniqueness and non-NULL requirements, you can reapply the primary key constraint.

Foreign key from table Enrollments to table Students :

1. In table Enrollments, insert a new tuple with a NULL student ID. Next, insert a new tuple with a student ID that is not listed in table Students. Observe the error messages that you get.

We got the same error messages as before related to the primary key.

When we tried to insert a student ID who is not listed, the error message indicate that the foreign key constraint has been violated, and it will specify that the referenced key (student ID) does not exist in the referenced table (Students).

2. Drop the foreign constraint from table Enrollments to table Students and perform again the previous two inserts. Explain what you observe.

Again, dropping a constraint will allow us to insert tuples without checking referential integrity. So they are no error messages, even if we insert NULL or non-existent student ID's.

3. Try to restore the foreign key constraint you just dopped. Explain what you observe.

if we have a tuples in Table Enrollments with NULL or non-existent student ID's, we will receive the same error messages indicating that the foreign key constraint violation. So again, when we try to restore the foreign key constraint the database will enforce referential integrity again.

4. Modify the instance of the table so that you can now restore the foreign key constraint.

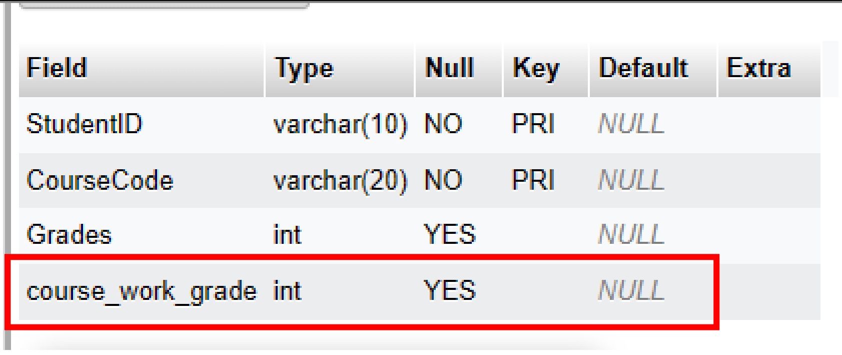
Same as before related to the primary key constraint. We need to ensure that all the "StudentID" column in the "Enrollment" table exist in Students table.

8 Modifying the Database's Schema

To record the course-work grade of each student for each course they are enrolled in, we should add this new attribute to the "Enrollment" table.

Using "ALTER TABLE" command, we don't need to drop and re-create the table from scratch.

```
-- Add a new attribute named "course_work_grade" to the EnrollmentsAndGrades table
ALTER TABLE EnrollmentsAndGrades
ADD course_work_grade INT;

-- Display the updated EnrollmentsAndGrades table structure
DESCRIBE EnrollmentsAndGrades;
```



Field	Type	Null	Key	Default	Extra
StudentID	varchar(10)	NO	PRI	NULL	
CourseCode	varchar(20)	NO	PRI	NULL	
Grades	int	YES		NULL	
course_work_grade	int	YES		NULL	

If we add a new column and we don't specify the default value. The default value of the new attribute "course_work_grade" will be NULL, unless if we had configured, like below :

```
-- Add a new attribute with a default value of 0 to the EnrollmentsAndGrades table
ALTER TABLE EnrollmentsAndGrades
ADD course_work_grade INT DEFAULT 0;
```

Conclusion

In this lab, we discovered the design and development of a comprehensive database. Through this exercise, several key aspects of database management were addressed :

- Schema Design : We carefully structured a database for school students.
- SQL script creation : We created SQL script for initializing and populating our database.
- Modification and Constraint : We explored different scenarios involving insertion of new records, change existing records, and impact of primary and foreign keys.
- Schema evolution : Instead to re-create, we used "ALTER TABLE" statement to extend our table.

Overall, this lab exercise offered a practical understanding of the fundamentals of designing, developing, and modifying a relational database model to efficiently manage school-related database. It shows us the importance of maintaining data integrity through the careful implementation of constraints.