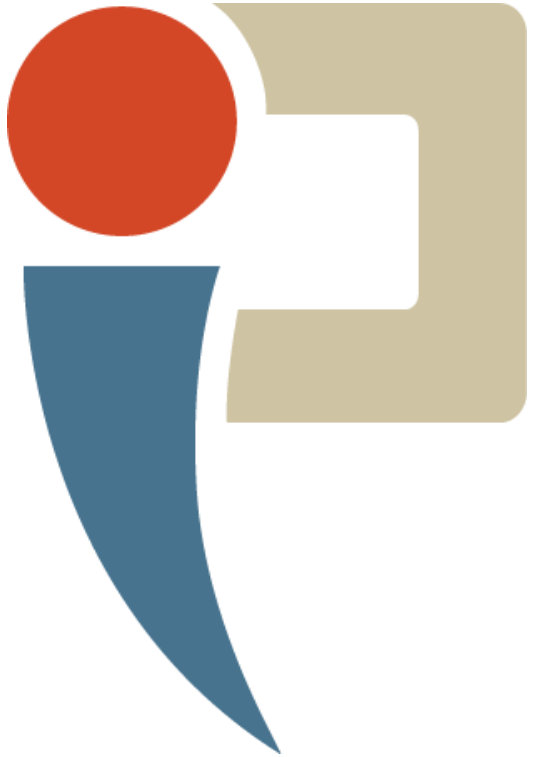
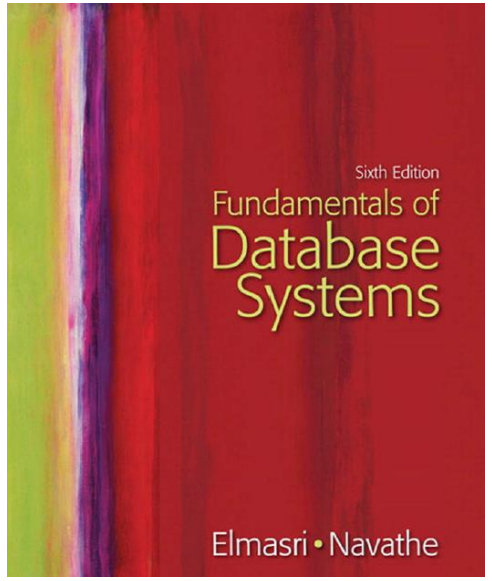


POLUDO INSTITUTE OF TECHNOLOGY & MEDIA

Instructor: Jesse Silva

Where can we find help?

- Books
  - **Fundamentals of database systems – 6th edition**
  - **Head First SQL - Lynn Beighley**



- Internet
  - W3Schools

    http://www.w3schools.com/sql
  - http://learndatamodeling.com/blog/creating-a-physical-data-model-using-erwin
  - http://learndatamodeling.com/blog/erwin-tutorial/
  - http://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561

What will we see here?

- Modelling the database
- Creating tables
- Applying the constraints
- Creating table relationships

FROM OUR LAST CHALLENGE:

Design a set of tables that can store information about a student, his personal information, where he lives, his courses, his grades, his tuition fees and the total duration of his courses. (everything you think is necessary to know about a student)

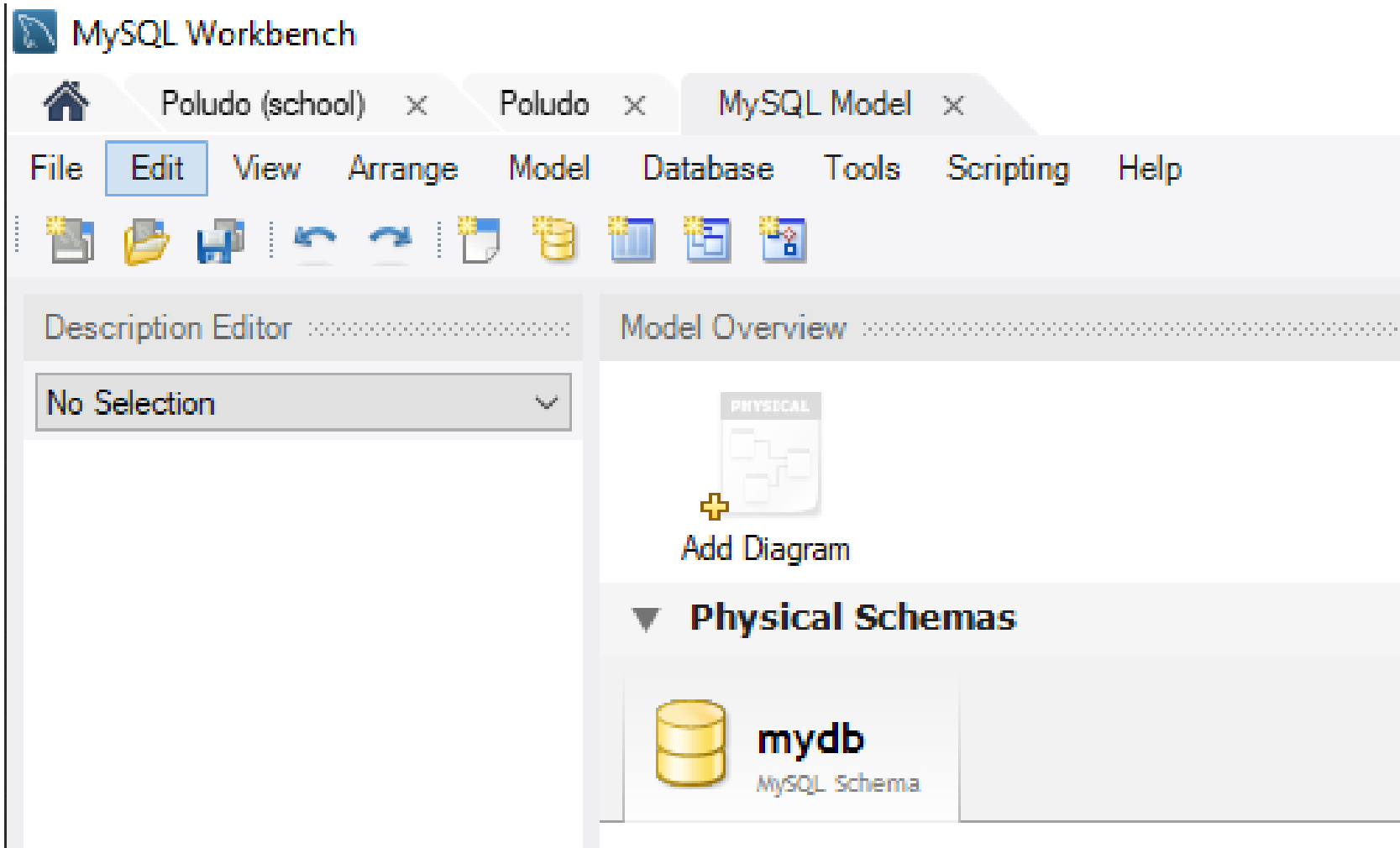
Step 1 - Modeling the database

- One Student can be enrolled in zero or more courses
- One course has zero or more students enrolled.
- A course is composed by one or more classes.
- A class can belong to one or more courses.
- A class can have only one instructor.
- A instructor can teach in more than one class.
- A class can have more than one schedule.
- The tuition is related with a class and changes every year
- A class can have one or more tuition
- One schedule belongs only to one class.
- One address belongs only to one student or instructor

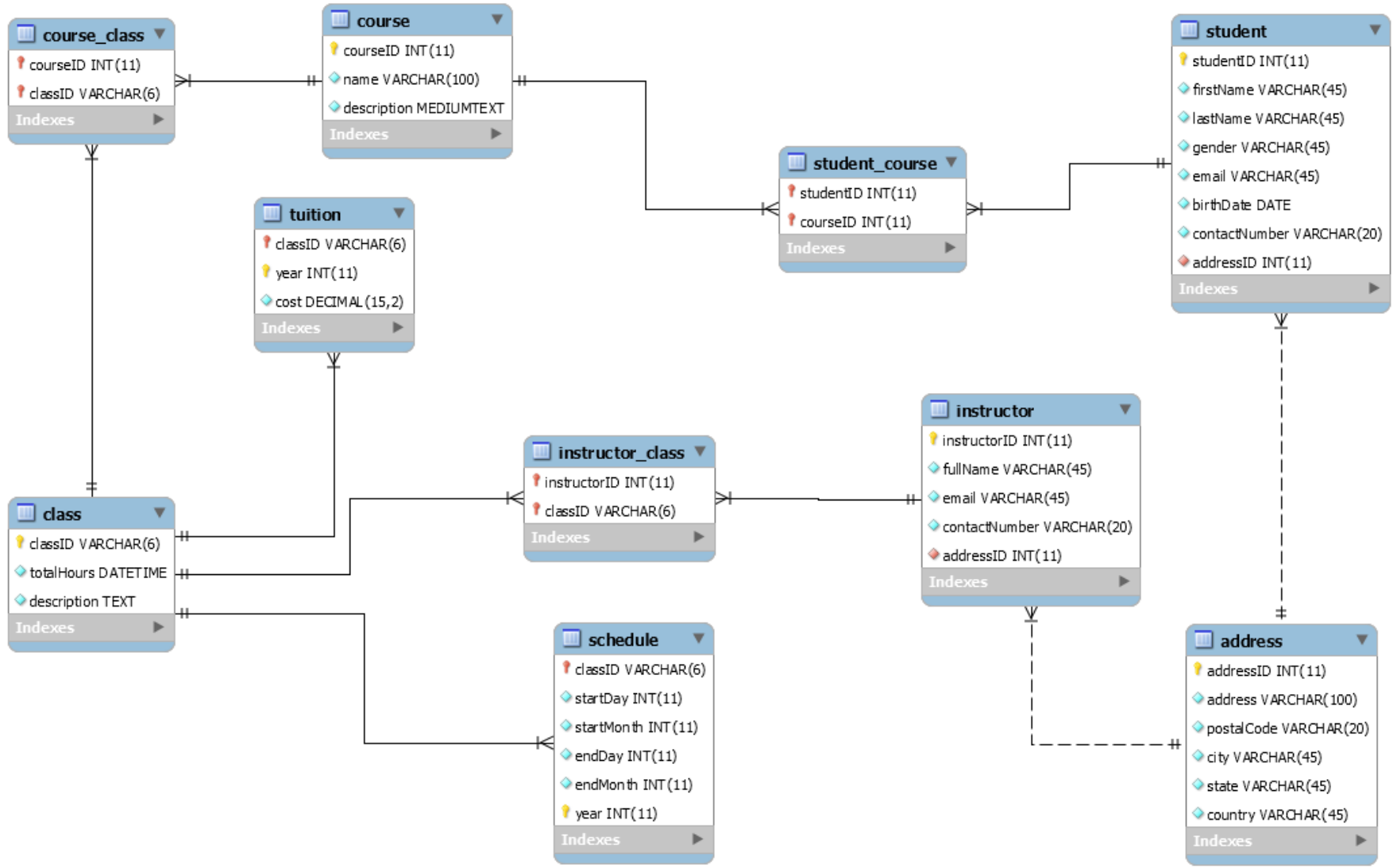
Step 1 - Modeling the database

We can use several programs to model our database and tables. One example is the MySQL Workbench Model feature.

THE INSTRUCTOR WILL SHOW YOU HOW TO ACCESS IT



Step 1 - Modeling the database



Step 2 – Creating the tables

```
CREATE TABLE TableName
            (
            columnName1 data_type(size) constraintName,
            columnName2 data_type(size) constraintName,
            columnName3 data_type(size) constraintName,
            ....
            );
```

Step 2 – Creating the tables

```
CREATE TABLE TableName
        (
        columnName1 data_type(size) constraintName,
        columnName2 data_type(size) constraintName,
        columnName3 data_type(size) constraintName,
        ....
        );
```

- ColumnName: specify the names of the table columns. Each column name must be unique inside the same table.
- Data_type: specifies the type of data the column can hold (e.g. varchar, integer, decimal, date, etc.).
- Size: define the exact or maximum length of the column.

Step 2 – Creating the tables

```
CREATE TABLE TableName
           (
           columnName1 data_type(size) constraintName,
           columnName2 data_type(size) constraintName,
           columnName3 data_type(size) constraintName,
           ....
           );
```

- ConstraintName: As described in the module 2, a constraint defines some attributes for a given column:

- ✓ NOT NULL - Indicates that a column cannot store NULL values
- ✓ UNIQUE - Ensures that each row for a column must have a unique value
- ✓ PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- ✓ FOREIGN KEY - Ensure the referential integrity of the data in one table to match values in another table
- ✓ CHECK - Ensures that the value in a column meets a specific condition
- ✓ DEFAULT - Specifies a default value for a column

Our first table: Student

Table Name: student

Collation: utf8 - default collation

Comments: new students table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
studentID	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
firstName	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
lastName	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
gender	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
email	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
birthDate	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
contactNumber	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
address	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	

Our first table: Student

```
CREATE TABLE `student` (
  `studentID` int(11) NOT NULL AUTO_INCREMENT,
  `firstName` varchar(45) NOT NULL,
  `lastName` varchar(45) NOT NULL,
  `gender` varchar(45) NOT NULL COMMENT 'Considering more than 2 genders',
  `email` varchar(45) NOT NULL,
  `birthDate` date NOT NULL,
  `contactNumber` varchar(20) NOT NULL,
  `address` varchar(255) NOT NULL,,
  PRIMARY KEY (`studentID`),
  KEY `addressFK_idx` (`address`),
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='new students table';
```

CREATING THE RELATIONSHIPS

Before creating our other tables, we need to learn how to connect them, meaning that we need to understand their relationship. In SQL there are 3 relationship types:

One-to-one relationships

Each row in one table is linked to one and only one row in another table. In a one-to-one relationship between Table A and Table B, each row in Table A is linked to another row in Table B.

One-to-many relationships

Each row in one table can be related to many rows in the relating table. This effectively save storage as the related record does not need to be stored multiple times in the relating table.

Many-to-many relationships

One or more rows in a table can be related to 0, 1 or many rows in another table. In this kind of relationship, we need to use junction tables, also known as mapping tables.

CREATING THE RELATIONSHIPS

In order to make these relationships, we will make use of Foreign keys, but we need also to define rules about what happens with the child table when we change or delete a record in the parent table. Here are the rules you can apply:

SPECIFICATION	UPDATE ON PARENT	DELETE ON PARENT
NO ACTION	Not allowed. Error message would be generated.	
CASCADE	Associated values in child table would also be updated or deleted	
SET NULL	Associated values in child table would be set to NULL. Foreign key column should allow NULL values to specify this rule.	
SET DEFAULT	Associated values in child table would be set to default value specified in column definition. Also default value should be present in primary key column. Otherwise basic requirement of FK relation would fail and update/delete operation would not be successful. If no default value is provided in foreign key column this rule could not be implemented.	

*If nothing is specified then the default rule is No Action.

Our second table: Address

Now that we know the rules, and also know that our students need an address, lets create the table to hold it. The basic information in our address table will be:

Which one is the primary key?

Is this primary key auto generated? Does it need to be?

What is the datatype of each column?

How about the relation with the Student table?

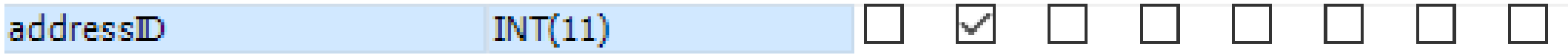
Remember that we created an address column in the student table and defined it as number instead of varchar?

We can use that column now to connect these two tables!

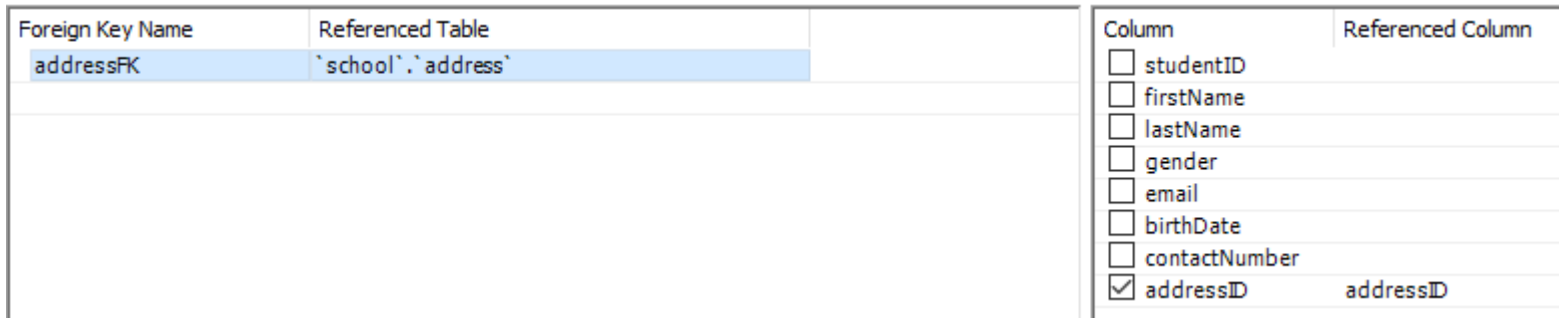
Open the student table in the SQL editor and follow the instructor on how to transform the address column into a foreign key apply the proper rules

How about the relation with the Student table?

Change the type and the name of the column



Go to the foreign keys tab and add the proper keys



Some useful commands to use on tables

- Removing a column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

- Adding a new column

```
ALTER TABLE table_name
ADD column_name datatype;
```

- Changing a column data type

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

CREATE A DUMMY TABLE IN YOUR SYSTEM AND TEST EACH OF THESE COMMANDS AND DISCUSS THE RESULTS

Some useful commands to use on tables

- Deleting a table

```
DROP TABLE table_name;
```

We are not allowed to drop a table that is referenced by foreign key constraint. If we try, as example, to remove the table Instructor we will see the following message:

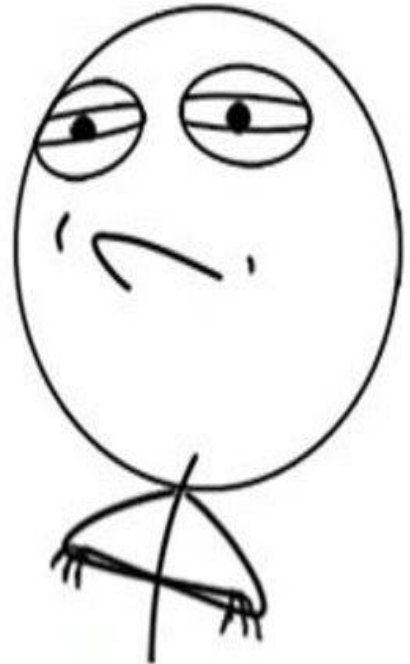
```
Error Code: 1217. Cannot delete or update a parent row: a foreign key constraint fails
```

In this situation we need first drop the foreign constraint and then drop the table.

CHALLENGE OF THE WEEK

- Now that we know how to create table relationships, we can go further and create all our tables.
- Be aware that some of the tables, as we have seen in the model will need to use junctions tables.
  - Research a little bit more on how to use them!

1. Course
2. Class
3. Instructor
4. Tuition
5. schedule



SOMETHING TO THINK ABOUT

- Analyze the created database and tables and try to see if there is a way of making them even better,
  - Check if they respect the normalization forms learned in the last module
  - Can you improve this database in any way?
    - Bring your ideas to the next class and lets discuss them