WILD CODE SCHOOL

Les Bases de données

Modélisation


@facez

Comment concevoir / modéliser une BDD

Comment penser l'architecture de ma BDD ?
Comment faire pour que mes données soient bien organisées ?

Utilisation de la méthode MERISE

- Méthode de conception, de développement et de réalisation de projets informatiques
- Créée dans les années 70, très utilisée en France
- Application à la modélisation de BDD
- Séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

Les modèles

- **Modèle Conceptuel de données (MCD)**

Permet d'établir une représentation claire des données et définit les dépendances fonctionnelles de ces données entre elles.

- **Modèle Logique de données (MLD)**

Traduction du modèle conceptuel de données en y ajoutant notamment les clés primaires et clés étrangères

- **Modèle Physique de données (MPD)**

Représentation finale d'une base de données, prenant en compte les spécificités du SGBD utilisé

Exemple

une école de code

Nous souhaitons enregistrer les **élèves** inscrits dans des **écoles** qui enseignent différents **langages**

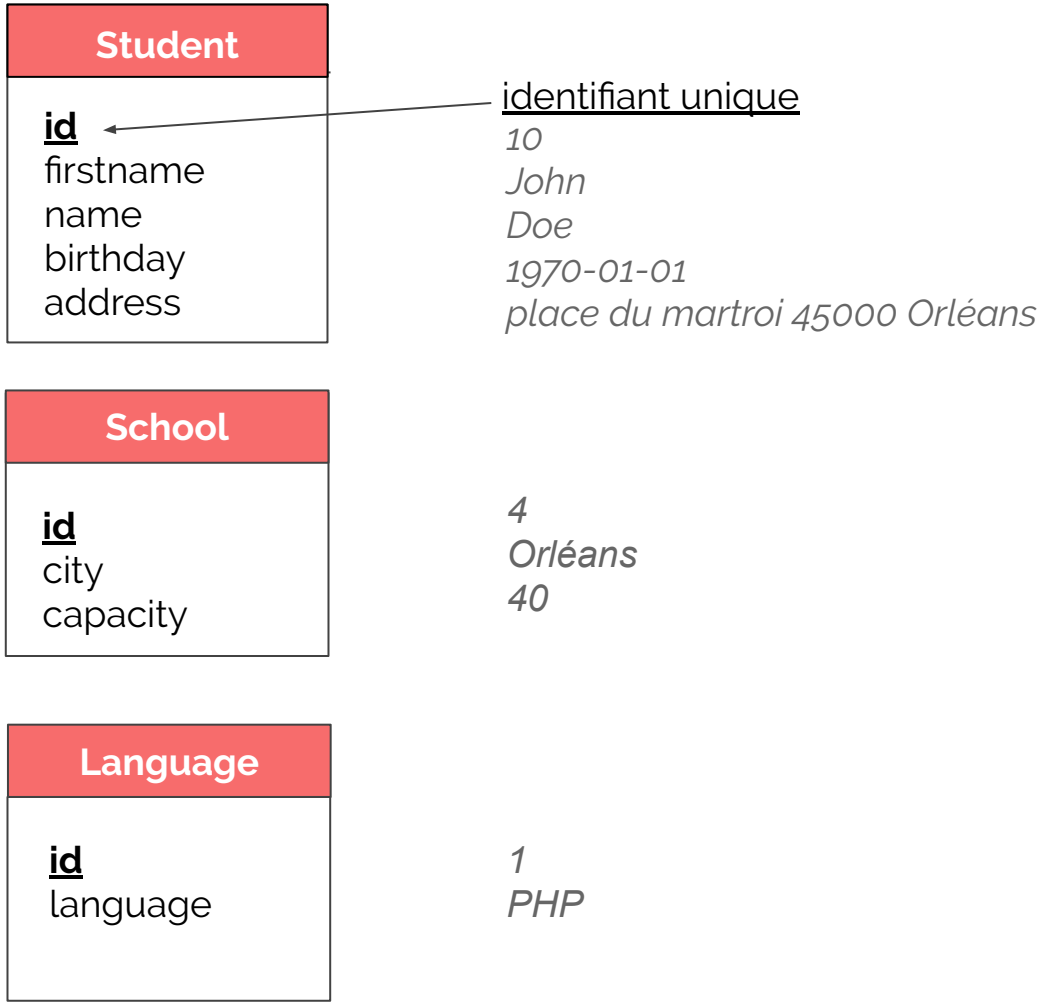
Comment organiser ces informations ?
Comment créer les tables et les champs correspondants ?

Les entités

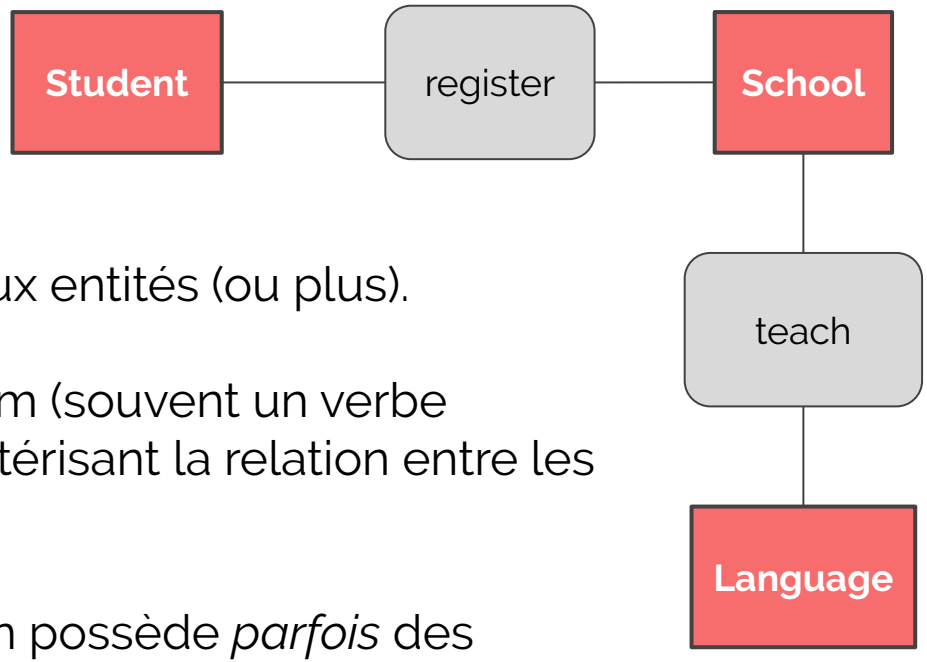
Une entité est un regroupement d'*éléments* ayant les mêmes caractéristiques.

Une entité possède des propriétés permettant de caractériser celle-ci.

Une entité a un identifiant unique

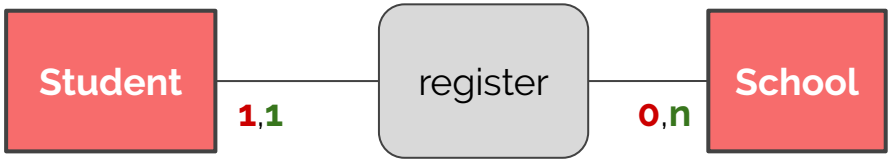


Les relations ou associations



- **Liens** entre deux entités (ou plus).
- Elles ont un nom (souvent un verbe d'action), caractérisant la relation entre les entités.
- Une association possède *parfois* des propriétés.

Les cardinalités



Un `Student` **est inscrit** dans **1 et 1 seule** `School`
(Dans) une `School` **sont inscrits** **0 à n** `Student`

- Indication du nombre **min** et **max** de liens entre 2 entités
- Pour une association de 2 entités, il y a donc 4 cardinalités à indiquer.
- 3 valeurs typiques : 0, 1 et n (plusieurs)

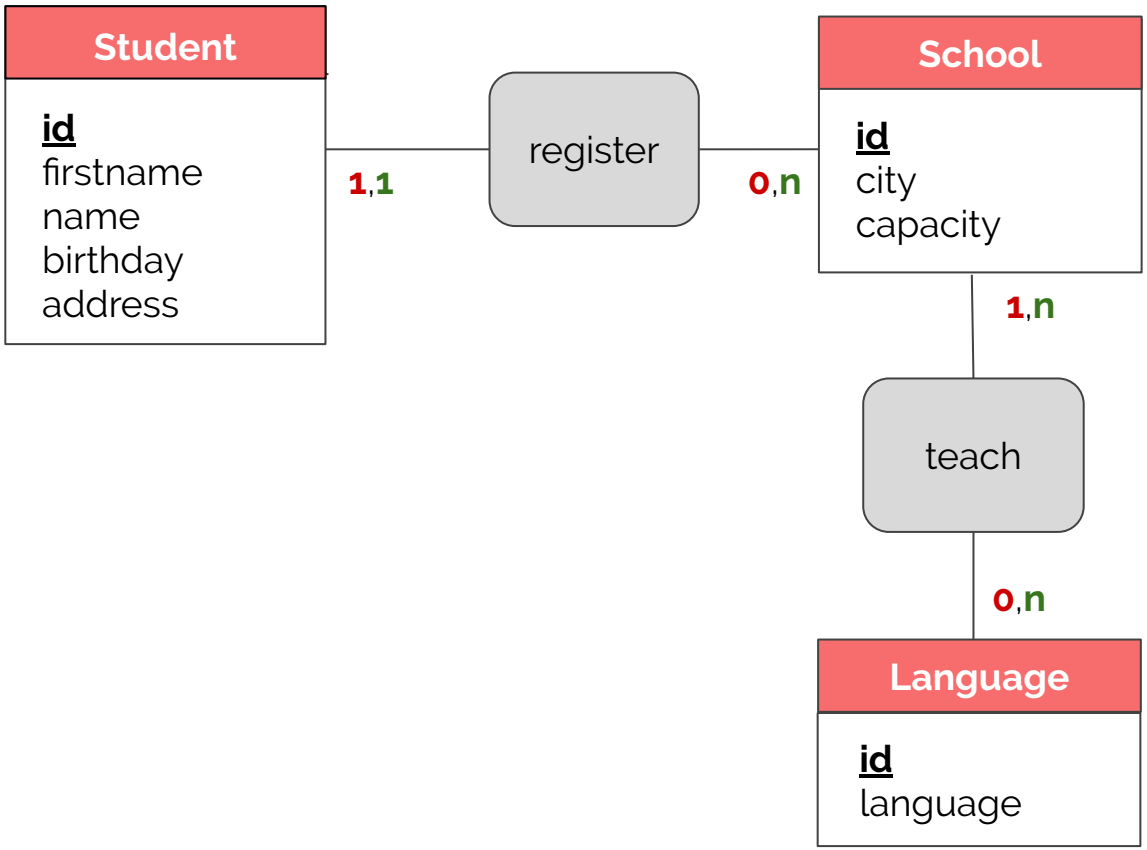
Les cardinalités

Une `School` **enseigne 1 à n** `Language`
Un `Language` **est enseigné dans 0 à n** `School`

- Indication du nombre **min** et **max** de liens entre 2 entités
- Pour une association de 2 entités, il y a donc 4 cardinalités à indiquer.
- 3 valeurs typiques : 0, 1 et n (plusieurs)

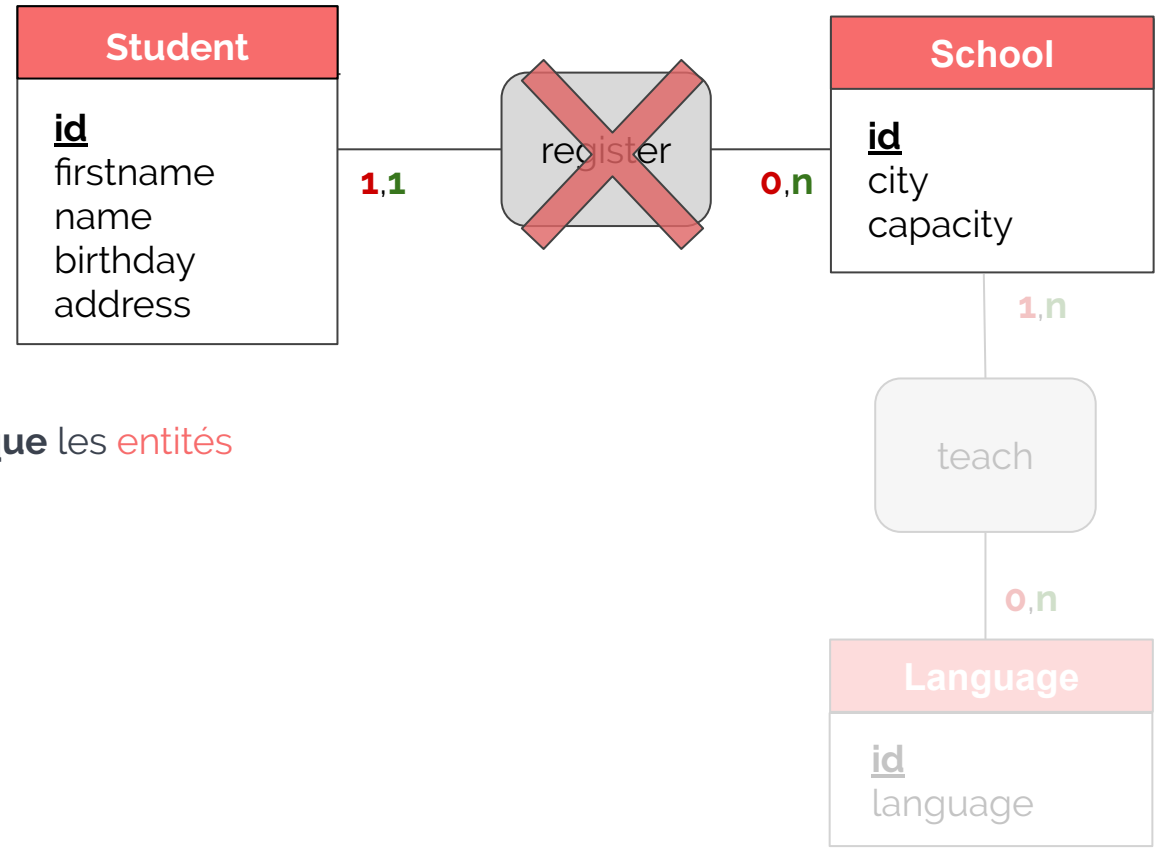
Le MCD

Ajout des champs



Passage MCD > MLD

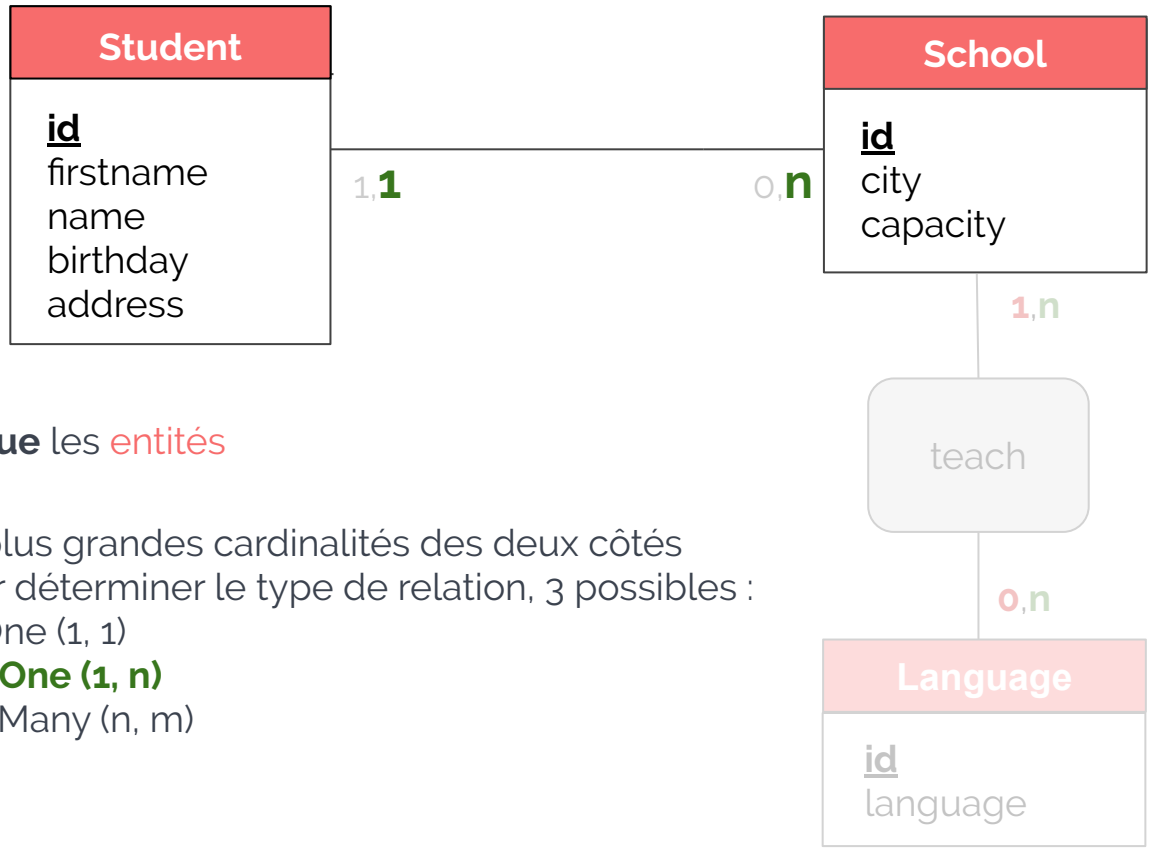
Cas d'une relation Many-To-One (1-n)



On ne garde **que** les entités

Passage MCD > MLD

Cas d'une relation Many-To-One (1-n)



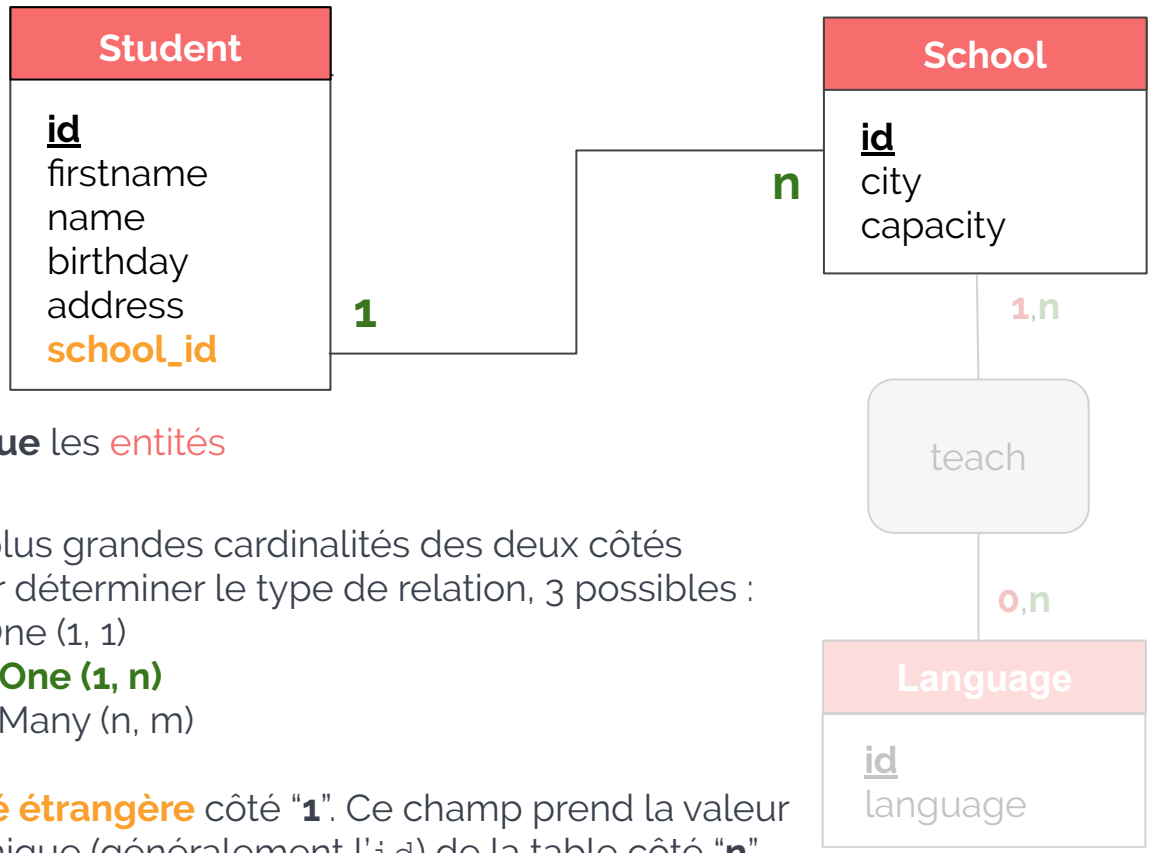
On ne garde **que** les entités

On prend les plus grandes cardinalités des deux côtés **(ici 1 et n)** pour déterminer le type de relation, 3 possibles :

- One To One (1, 1)
- **Many To One (1, n)**
- Many To Many (n, m)

Passage MCD > MLD

Cas d'une relation Many-To-One (1-n)



On ne garde **que** les entités

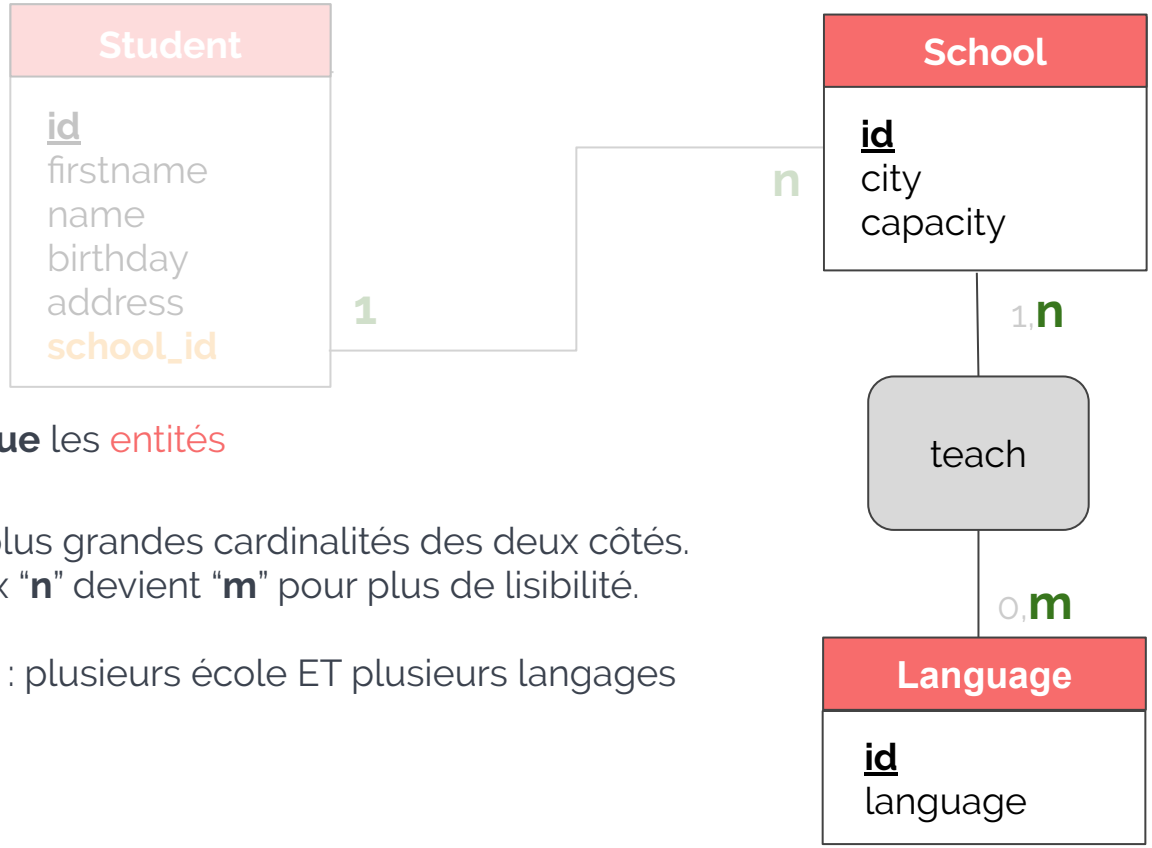
On prend les plus grandes cardinalités des deux côtés **(ici 1 et n)** pour déterminer le type de relation, 3 possibles :

- One To One (1, 1)
- **Many To One (1, n)**
- Many To Many (n, m)

Ajout d'une **clé étrangère** côté "**1**". Ce champ prend la valeur d'un champ unique (généralement l'`id`) de la table côté "**n**"

Passage MCD > MLD

Cas d'une relation Many-To-Many (n-m)



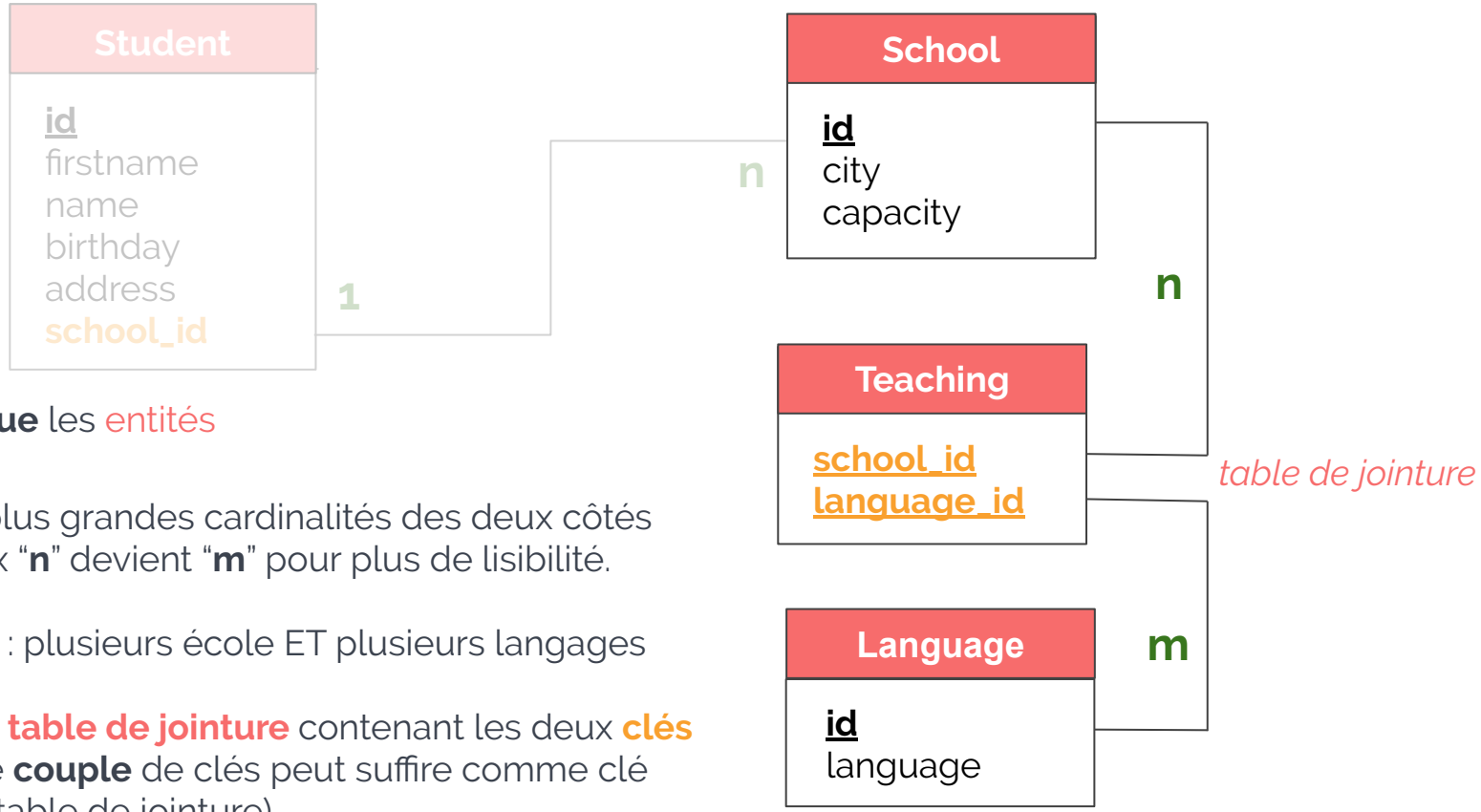
On ne garde **que** les entités

On prend les plus grandes cardinalités des deux côtés.
L'une des deux "**n**" devient "**m**" pour plus de lisibilité.

Many To Many : plusieurs école ET plusieurs langages

Passage MCD > MLD

Cas d'une relation Many-To-Many (n-m)



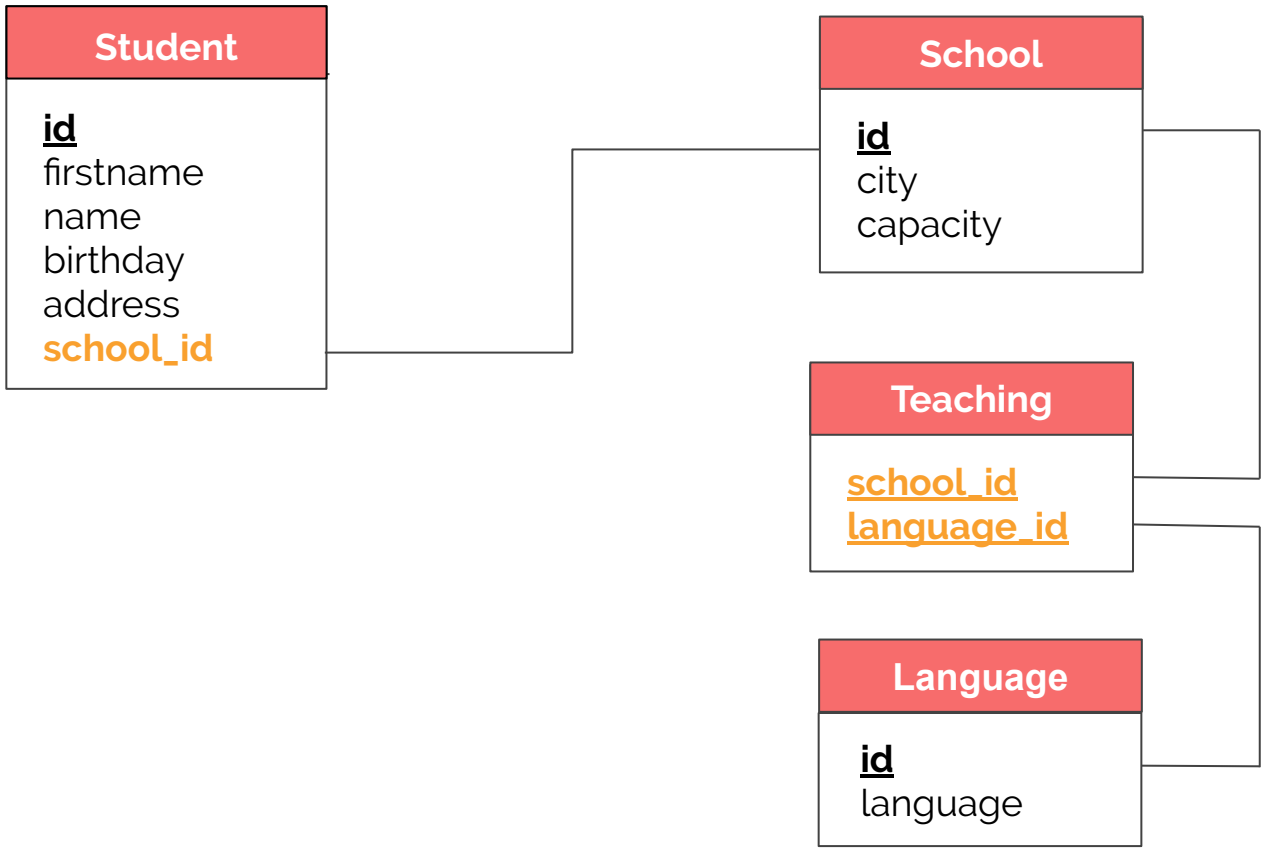
On ne garde **que** les entités

On prend les plus grandes cardinalités des deux côtés
L'une des deux "**n**" devient "**m**" pour plus de lisibilité.

Many To Many : plusieurs école ET plusieurs langages

Création d'une **table de jointure** contenant les deux **clés étrangères** (ce **couple** de clés peut suffire comme clé primaire de la table de jointure)

Le MLD

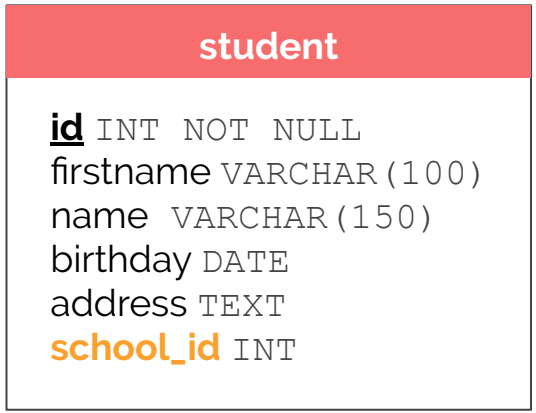

Student
id
firstname
name
birthday
address
school_id
School
id
city
capacity
Teaching
school_id
language_id
Language
id
language

Le MPD

Modèle Physique de données

Le niveau **physique** tient compte des particularités de chaque SGBDR

- Types des données (INT, VARCHAR, CHAR, BOOL…)
- Contraintes (unique, nullable, auto-incrémentation…)
- indexes (amélioration des performances) ...



Contraintes d'intégrité

Règles à suivre quand des enregistrements de tables reliées sont **mis à jour** ou **supprimés**

Ex : J'efface une école de la table `school`, reliée à des élèves de la table `student`

- si aucune contrainte n'est définie, *l'école* est **effacée** et les *élèves* reliés à cette école se retrouvent alors "orphelins"
- si une contrainte est définie **sans option**, la suppression est **refusée** car des *élèves* sont associés à cette *école* (par contre, on pourra effacer un élève qui lui n'est relié qu'à une et une seule école !)
- si une option **CASCADE** est définie, les *élèves* associés à *école* seront automatiquement **supprimés**

```
mysql> CREATE TABLE student
    ...
  PRIMARY_KEY(id)
  FOREIGN KEY (school_id)
    REFERENCES school(id)
    ON DELETE CASCADE
    ON UPDATE NO ACTION;
```

Quels outils ?

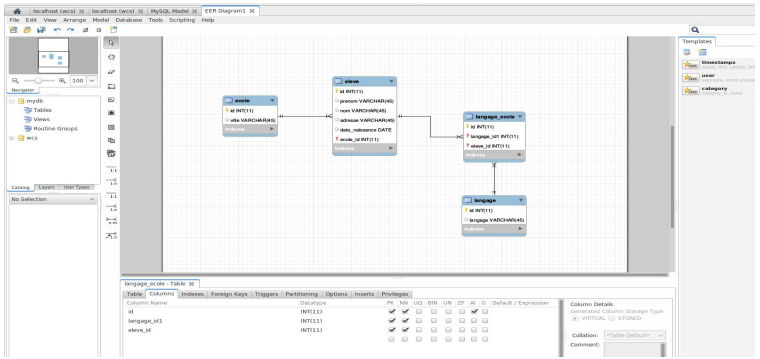
Papier et crayon

Rapide, efficace, pour un premier “jet”

Interfaces graphiques

MySQL Workbench (ou PhpStorm, PhpMyAdmin)...

```
sudo apt install mysql-workbench
```



Complet, possibilité de générer facilement
le SQL correspondant aux tables modélisées

Ressources

Tutoriel vidéo sur la méthode Merise

http://ineumann.developpez.com/tutoriels/merise/initiation-merise/

Série de vidéos sur JMerise