WILD CODE SCHOOL

Relations SQL

Base de données relationnelle

Modélisation

Méthode Merise

MERISE

La méthode Merise est une méthodologie française de conception de systèmes d'information.

Elle se base sur plusieurs modèles :

- **Modèle Conceptuel de données (MCD)**
  Représentation claire des données. Définition des dépendances fonctionnelles.

- **Modèle Logique de données (MLD)**
  Ajout des clés primaires et étrangères

- **Modèle Physique de données (MPD)**
  Prise en compte des spécificités du SGBD utilisé

Source

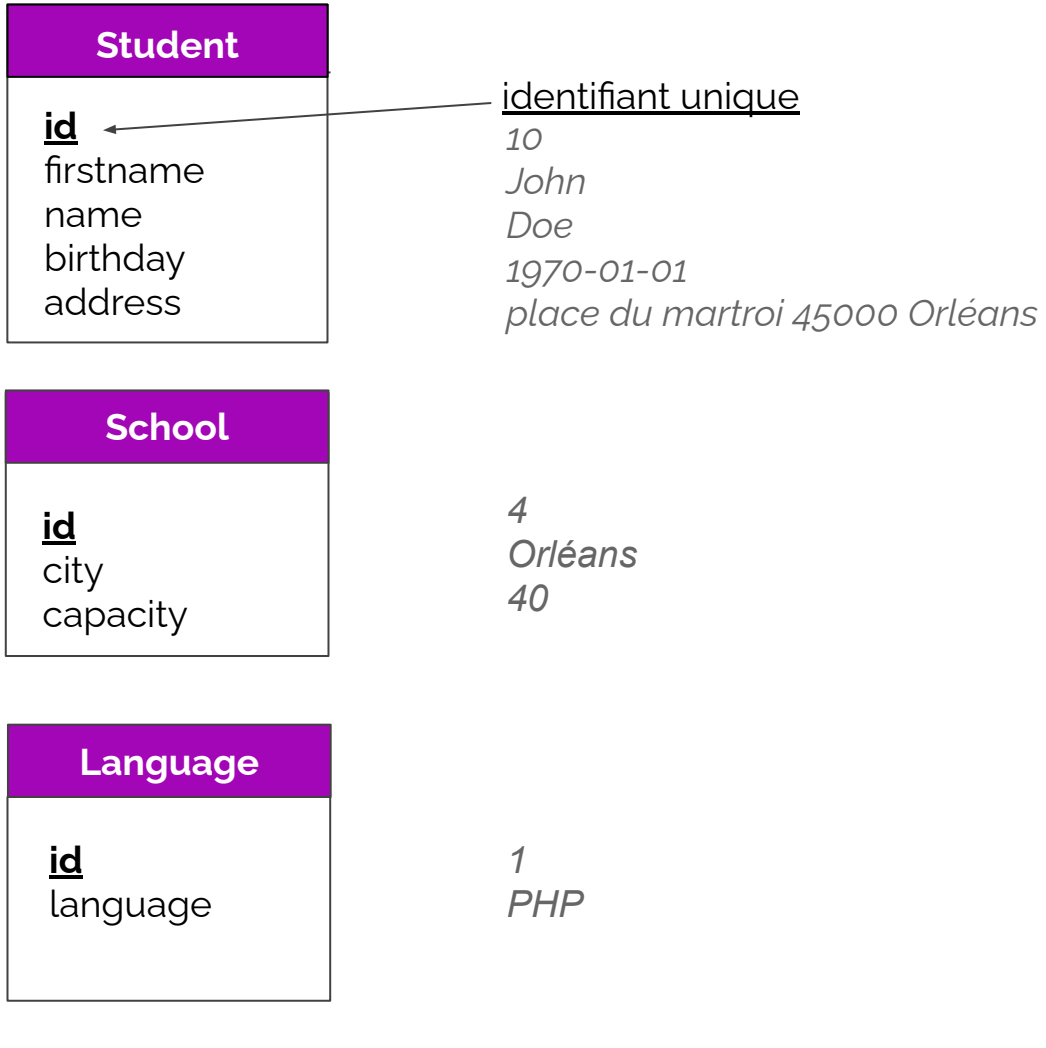
Les entités

Représentée par une table.

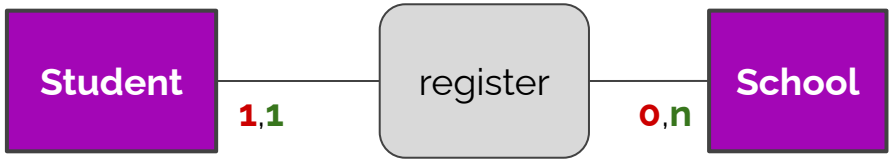
Regroupement d'informations.

Possède des propriétés (caractéristiques).

Identifiant unique



Les cardinalités

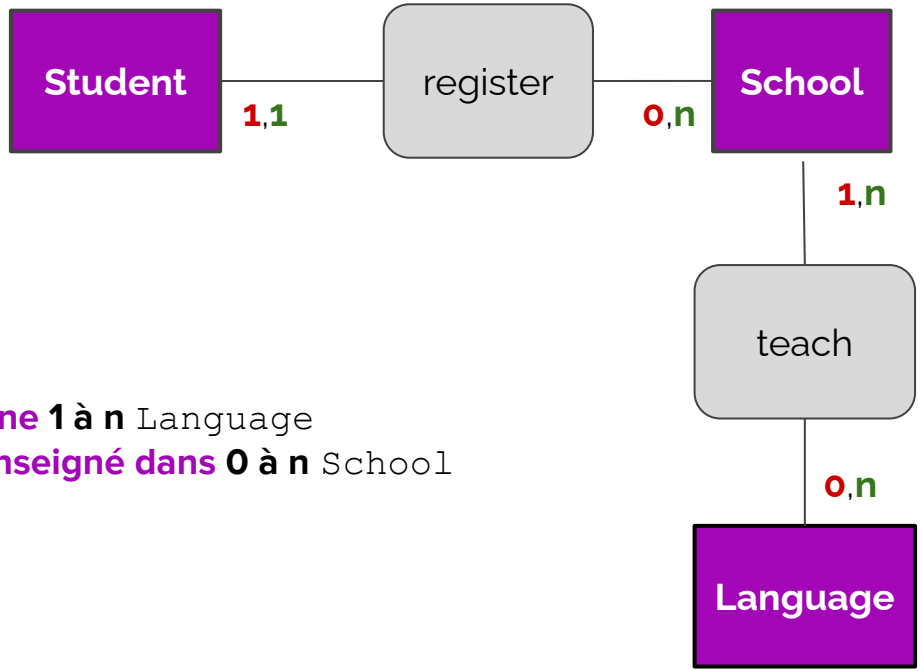


Un `Student` **est inscrit** dans **1 et 1 seule** `School`
(Dans) une `School` **sont inscrits 0 à n** `Student`

- 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)

Source

Les cardinalités



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

Source

Les différents types de relations

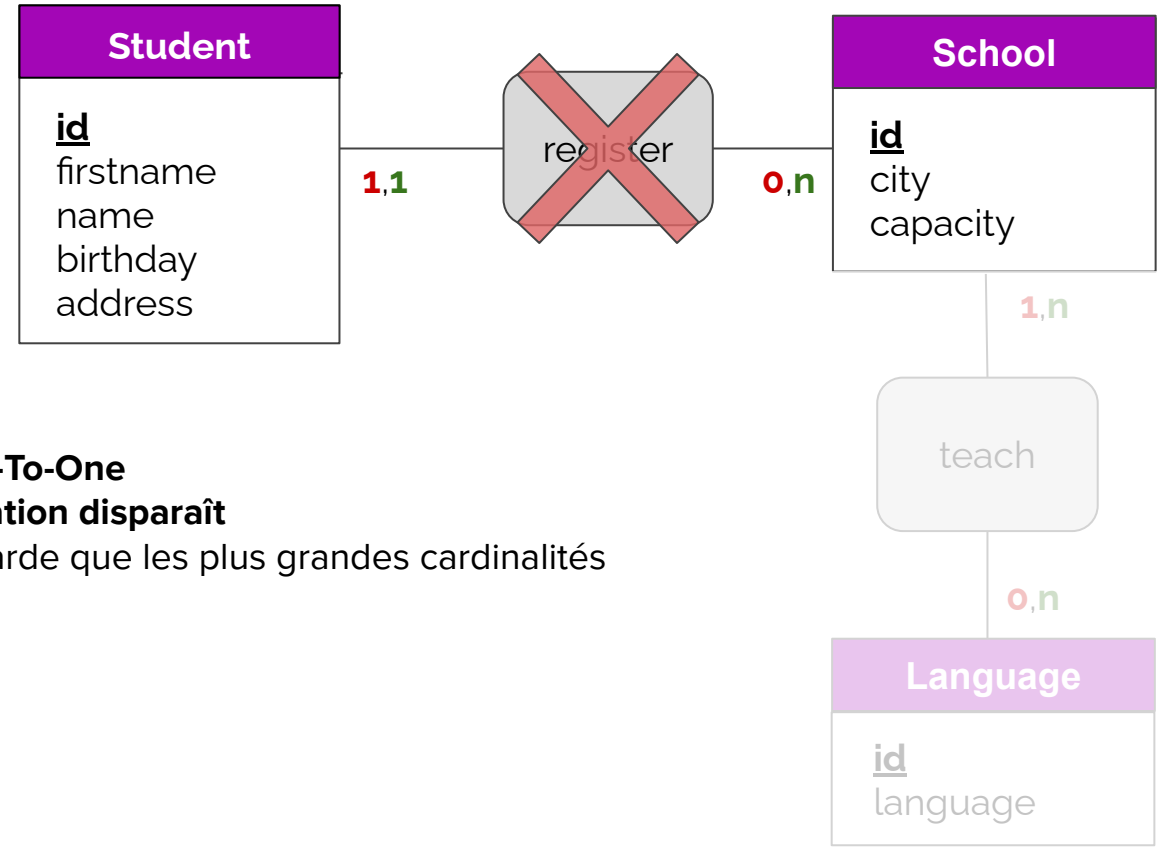
- Pour déterminer le type d'une relation, prendre les plus grandes cardinalités des deux côtés de la relation.

- Le passage du MCD au MLD est régi par un **ensemble de règles** pour chaque relations.

- Il existe 3 types de relations :
  - **One To One** (1, 1)
  - **Many To One** (1, n)
  - **Many To Many** (n, m)

Passage du MCD => MLD

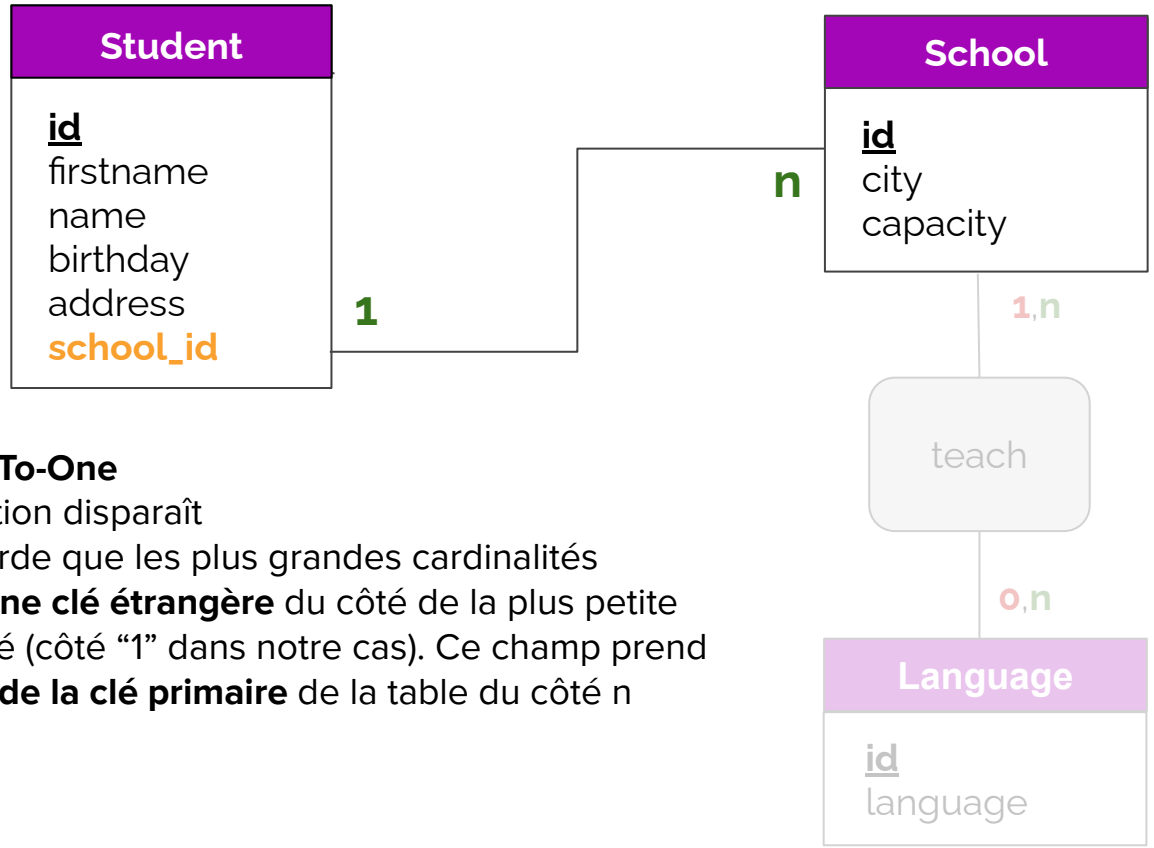
Cas d'une relation Many-To-One (1-n)



- Relation **Many-To-One**
  - **L'association disparaît**
  - On ne garde que les plus grandes cardinalités

Passage du MCD => MLD

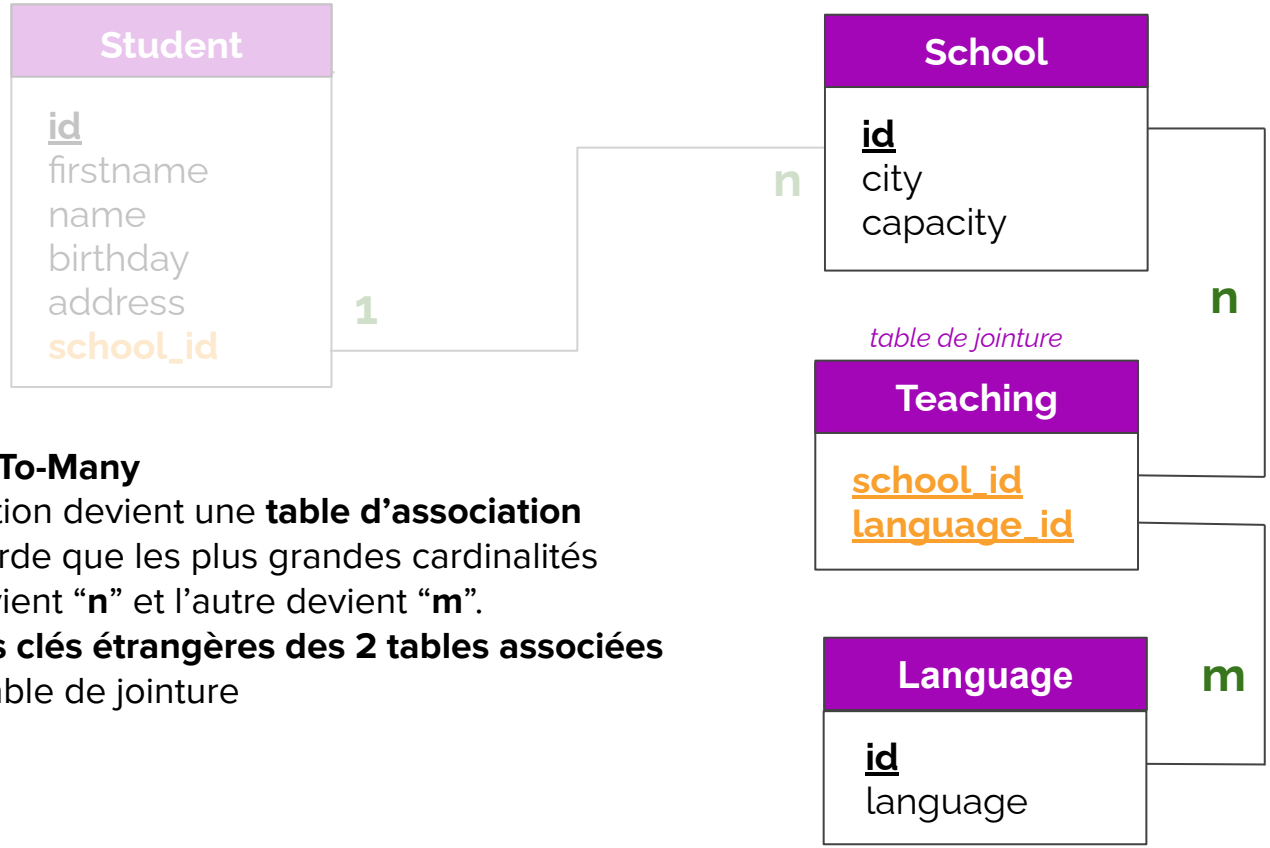
Cas d'une relation Many-To-One (1-n)



- Relation **Many-To-One**
  - L'association disparaît
  - On ne garde que les plus grandes cardinalités
  - **Ajout d'une clé étrangère** du côté de la plus petite cardinalité (côté "1" dans notre cas). Ce champ prend la **valeur de la clé primaire** de la table du côté n

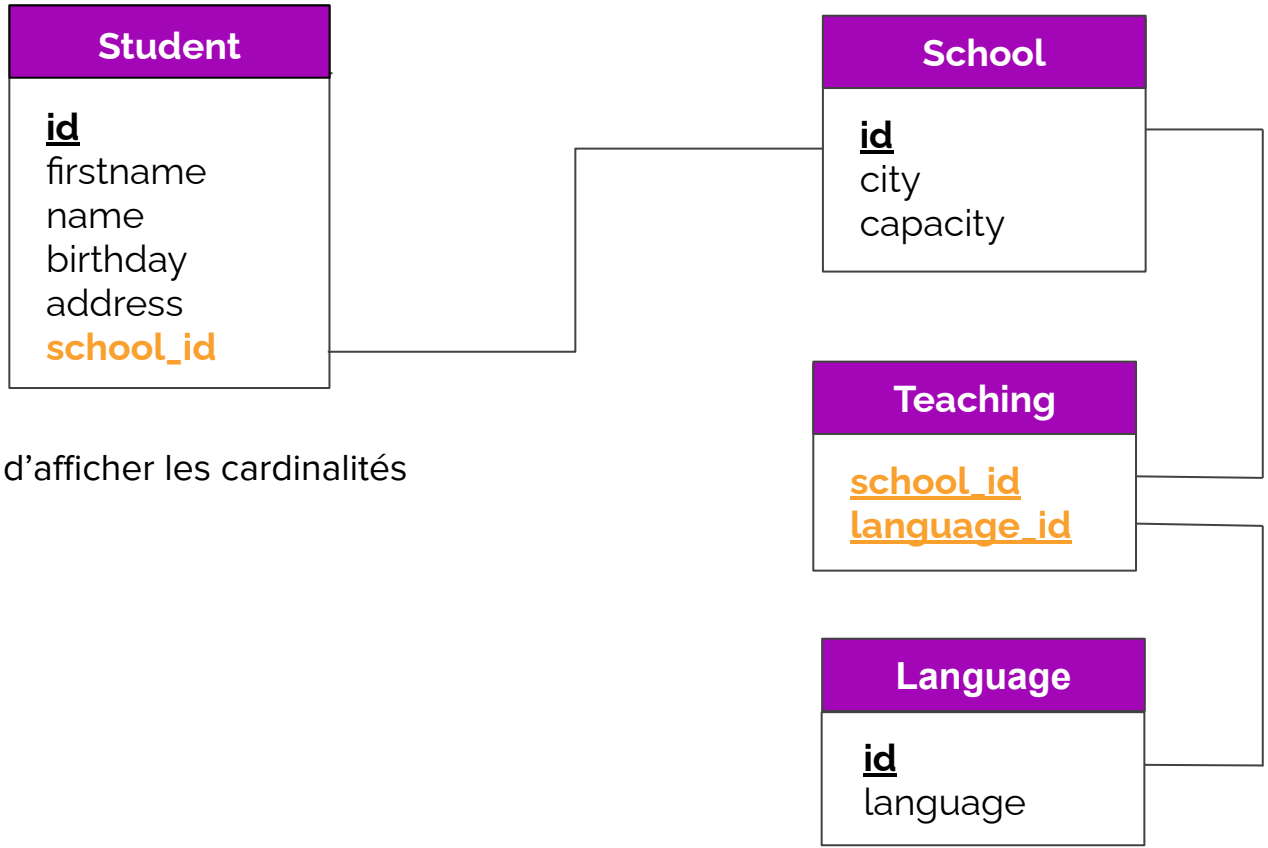
Passage du MCD => MLD

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



- Relation **Many-To-Many**
  - L'association devient une **table d'association**
  - On ne garde que les plus grandes cardinalités
    L'une devient "**n**" et l'autre devient "**m**".
  - **Ajout des clés étrangères des 2 tables associées** dans la table de jointure

Le Modèle Logique de Données - MLD



- Pas nécessaire d'afficher les cardinalités

Le Modèle Physique de Données - MPD

- Prise en compte des particularité de chaque SGBDR
- Types de données : chaque SGBDR à ses propres types de données
- Contraintes : unique, nullable, auto-incrément...
- Index : performances

student
id `INT NOT NULL` firstname `VARCHAR(100)` name `VARCHAR(150)` birthday `DATE` address `TEXT` **school_id** `INT`

SQL avancé

On fait des requêtes ?

Clés étrangères

- Définition de la clé étrangère **lors la création de la table**
- Une clé étrangère est une **contrainte d'intégrité**

```
mysql> CREATE TABLE student (
      …,
      school_id INT,
      FOREIGN KEY (school_id) REFERENCES school(id)
);
```

Clés étrangères

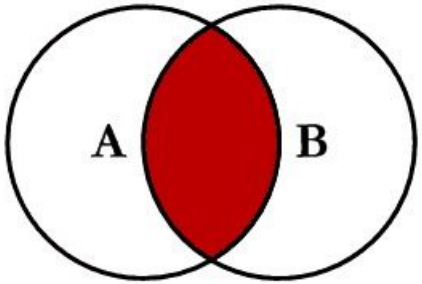
- La table existe déjà : **ALTER**
  - Ajout de la colonne
  - Ajout de la nouvelle contrainte d'intégrité

```
mysql> ALTER TABLE student
     ADD COLUMN school_id INTEGER,
     ADD CONSTRAINT FK_student
     FOREIGN KEY (school) REFERENCES school(id);
```

Les jointures

- Les **jointures** permettent de faire des requêtes en **associant** deux tables entre elles en fonction de champs **en commun**

```
INNER JOIN <table> ON <condition>
```




```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Les jointures

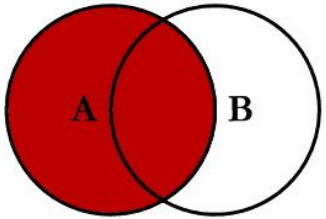
- **Exemple** : Sélection du nom, du prénom et également du nom de l'école pour les étudiants de la ville de Bordeaux

```
mysql> SELECT st.firstname, st.lastname, sc.city
       FROM student AS st
           INNER JOIN school AS sc ON sc.id = st.school_id
       WHERE sc.city = 'Bordeaux';

firstname |  lastname   |  city
----------+-------------+----------
Arthur    | Pendragon   | Bordeaux
Lancelot  | Du Lac      | Bordeaux
```

Les jointures

- Autres types de jointures :
  - **LEFT JOIN** : Retourne toutes lignes de la table ***FROM*** et les champs relatifs à la table ***JOIN*** - NULL si pas de correspondance




```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB
ON A.Key = B.Key
```

  - **RIGHT JOIN** : Retourne toutes les lignes de la table ***JOIN*** et les champs relatifs à la table ***FROM*** - NULL si pas de correspondance

Exemple : **https://www.w3schools.com/sql/sql_join_left.asp**

Les jointures

- **Exemple** : Elias est retourné même s'il n'a pas d'école attribué

```
mysql> SELECT firstname, lastname, city
       FROM student AS st
          LEFT JOIN school AS sc ON sc.id=st.school_id;

firstname | lastname      | city
----------+---------------+---------
Arthur    | Pendragon     | Bordeaux
Élias     | de Kelliwic'h | NULL
```

Aller plus loin

Index, Transaction, ACID

Indexes

- Basé sur **une ou plusieurs colonnes** d'une table
- **Amélioration des performances en lecture** mais **réduction** de la vitesse **en écriture**
- **Copie d'une table triée** dans l'ordre des colonnes demandé
- Mettre un index sur toutes les colonnes équivaut à ne pas avoir d'index et ralentira les écritures
- Il faut **trouver le bon équilibre** par rapport aux requêtes effectuées

Transactions

- Exécution d'un **ensemble de requêtes** d'un seul bloc
- Meilleur gestion des erreurs
  - Si une requête échoue on peut **annuler tout le bloc** (ou quand même valider les requêtes qui ont réussi)
- Gestion d'une transaction
  - Création : **START TRANSACTION**
  - Validation : **COMMIT**
  - Invalidation : **ROLLBACK**
  - Validation automatique : **autocommit=1** (par défaut sur MySQL)
    La validation se fait toute seule.

ACID

- Propriétés ACID
  - **Atomicité** : Toutes les opérations de la transaction sont exécutées dans leur intégralité ou aucune n’est exécutée
  - **Cohérence** : La transaction transforme la BDD d’un état cohérent à un autre état cohérent
  - **Isolation** : Les opérations d'une transaction sont isolées des autres transactions, ce qui signifie que les transactions en cours ne sont pas visibles de l'extérieur jusqu'à ce qu'elles soient terminées.
  - **Durabilité** : Une fois qu'une transaction a été confirmée, ses effets sont permanents et survivent aux pannes du système.

Atelier

The good corner

The Good Corner

Catégorisation des annonces.

Une annonce est liée à une catégorie.

Une catégorie à un nom.

Un internaute peut rechercher des annonces par catégorie.

Un internaute peut rechercher une annonce par un nom de catégorie.

Source

The Good Corner

- Créer les MCD, MLD et MPD
- Dans un fichier SQL
- Créer la table “**category**”
- Insérer ces catégories :
  - vêtement
  - voiture
  - autre
- Lier la table “**ad**” à la table “**category**”
- Ajouter une catégorie à chaque annonce
- Afficher les annonces de la catégorie “vêtement”
- Afficher les annonces des catégories “vêtement” et “voiture”
- Afficher le prix moyen des annonces de la catégorie “autre”
- Afficher les annonces des catégories dont le nom commence par un “v”