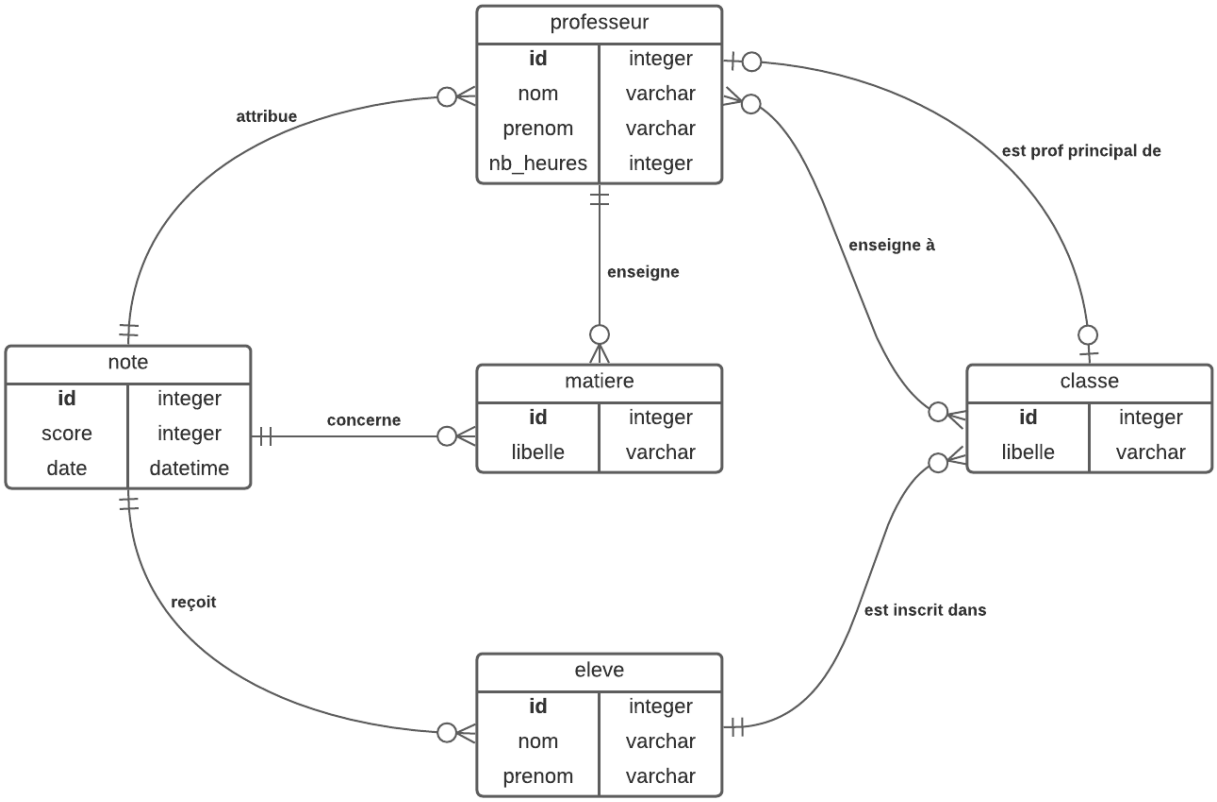
Ex Ecole - Lucien HAMM



Base de données

Tables

eleve

CHAMP	TYPE
id	integer
nom	varchar
prenom	varchar
classe_id	integer

professeur

CHAMP	TYPE

CHAMP	TYPE
id	integer
nom	varchar
prenom	varchar
nb_heures	integer

classe

CHAMP	TYPE
id	integer
libelle	varchar

enseigne

CHAMP	TYPE
professeur_id	integer
classe_id	integer

principal

CHAMP	TYPE
professeur_id	integer
classe_id	integer

matiere

CHAMP	TYPE
id	integer
libelle	varchar

note

CHAMP	TYPE
id	integer
score	integer
date	datetime
professeur_id	integer

CHAMP	TYPE
eleve_id	integer
matiere_id	integer

Relations

professeur - note

Un professeur attribue aucune ou plusieurs notes. (0,n)

Un note est attribuée par un seul professeur. (1,1)

professeur - matiere

Un professeur enseigne une seule matière. (1,1)

Une matière est enseignée par aucun ou plusieurs professeurs. (0,n)

professeur - classe

Un professeur enseigne à aucun ou plusieurs classes. (0,n)

Une classe a aucun ou plusieurs professeurs. (0,n)

Un professeur est le professeur principal d'aucune ou une seule classe. (0,1)

Une classe a aucun ou un seul professeur principal. (0,1)

matiere - note

Une matière est concernée par aucune ou plusieurs notes. (0,n)

Une note concerne une seule matière. (1,1)

eleve - note

Un élève reçoit aucune ou plusieurs notes. (0,n)

Une note concerne un seul et même élève. (1,1)

eleve - classe

Un élève est inscrit dans une seule classe. (1,1)

Une classe est rattachée à aucun ou plusieurs élèves. (0,n)

```sql
CREATE TABLE IF NOT EXISTS matiere (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    libelle VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS professeur (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL,
    nb_heures INTEGER NOT NULL,
    matiere_id INTEGER NOT NULL
);

CREATE TABLE IF NOT EXISTS classe (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    libelle VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS enseigne (
    professeur_id INTEGER,
    classe_id INTEGER,
    FOREIGN KEY (professeur_id) REFERENCES professeur (id),
    FOREIGN KEY (classe_id) REFERENCES classe (id)
);

CREATE TABLE IF NOT EXISTS principal (
    professeur_id INTEGER,
    classe_id INTEGER,
    FOREIGN KEY (professeur_id) REFERENCES professeur (id),
    FOREIGN KEY (classe_id) REFERENCES classe (id)
);

CREATE TABLE IF NOT EXISTS eleve (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL,
    classe_id INTEGER NOT NULL,
    FOREIGN KEY (classe_id) REFERENCES classe (id)
```

```
);

CREATE TABLE IF NOT EXISTS note (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    score INTEGER NOT NULL,
    `date` DATE NOT NULL,
    professeur_id INTEGER NOT NULL,
    eleve_id INTEGER NOT NULL,
    matiere_id INTEGER NOT NULL,
    FOREIGN KEY (professeur_id) REFERENCES professeur (id),
    FOREIGN KEY (eleve_id) REFERENCES eleve (id),
    FOREIGN KEY (matiere_id) REFERENCES matiere (id)
);
```

Ex 1

Afficher toutes les classes + professeur principal

```
SELECT c.*, pr.*
FROM classe c, professeur pr, principal pa
WHERE pr.id = pa.professeur_id
AND pa.classe_id = c.id
```

Ex 2

Afficher la liste des classes dont Virgile (id_prof = 33) est le professeur principal

```
SELECT c.*
FROM classe c, professeur pr, principal pa
WHERE c.id = pa.classe_id
AND pa.professeur_id = pr.id
AND pr.id = 33
```

Ex 3

Afficher la liste des professeurs qui enseignent les maths

```
SELECT p.*
FROM professeur p, matiere m
WHERE p.matiere_id = m.id
AND m.libelle = 'maths'
```

Ex 4

Afficher la liste des élèves de la classe '5eme B'

```
SELECT e.*
FROM eleve e, classe c
WHERE e.classe_id = c.id
AND c.libelle = '5eme B'
```

Ex 5

Afficher le nombre d'heures totales dispensées par le professeur Stéphane (id_prof = 12) toute classe confondue (on affichera le nom / prénom du professeur)

```
SELECT CONCAT(p.nom, ' ', p.prenom) as prof, SUM(p.nb_heures)
FROM professeur p, classe c, enseigne e
WHERE p.id = e.professeur_id
AND e.classe_id = c.id
AND p.id = 12
GROUP BY prof
```

Ex 6

Afficher le nombre d'heures totales dispensées par tous les professeurs toute classe confondue

```
SELECT p.*, SUM(p.nb_heures)
FROM professeur p, classe c, enseigne e
WHERE p.id = e.professeur_id
AND e.classe_id = c.id
GROUP BY p.id
```

Ex 7

Afficher la liste des classes totalisant un nombre d'élèves supérieur ou égal à 15

```
SELECT c.libelle, SUM(e.id)
FROM classe c, eleve e
WHERE e.classe_id = c.id
GROUP BY c.libelle
HAVING SUM(e.id) >= 15
```

Ex 8

Afficher les professeurs qui ne SONT PAS professeurs principaux

```sql
SELECT pr.*
FROM professeur pr, principal pa
WHERE pr.id = pa.professeur_id
AND pa.classe_id IS NULL
```

Ex 9

Afficher l'historique des notes de l'élève numéro 5 triées de la récente à la plus ancienne (avec la matière associée)

```sql
SELECT CONCAT(e.prenom, ' ', e.nom), n.score, m.libelle
FROM eleve e, note n, matiere m
WHERE e.note_id = n.id
AND n.matiere_id = m.id
AND e.id = 5
ORDER BY n.date DESC
```

Ex 10

Afficher la moyenne générale de chaque élève de la classe "3eme C" en classant la liste du meilleur élève au moins bon

```sql
SELECT CONCAT(e.prenom, ' ', e.nom), AVG(SUM(n.score))
FROM eleve e, note n, classe c
WHERE e.classe_id = c.id
AND n.eleve_id = n.id
AND c.libelle = '3eme C'
GROUP BY AVG(SUM(n.score))
ORDER BY AVG(SUM(n.score)) DESC
```

Ex 11

Afficher les 3 meilleurs élèves de la classe "3eme C" (avec leur moyenne respective)

```sql
SELECT CONCAT(e.prenom, ' ', e.nom), AVG(SUM(n.score))
FROM eleve e, note n, classe c
WHERE e.classe_id = c.id
AND n.eleve_id = n.id
AND c.libelle = '3eme C'
GROUP BY AVG(SUM(n.score))
ORDER BY AVG(SUM(n.score)) DESC
LIMIT 3
```

Ex 12

Insérer un nouvel élève dans la classe "4eme D" (id_classe = 5) : "Micka M"

```sql
INSERT INTO eleve(nom, prenom, classe_id) VALUES('M', 'Micka', 5)
```

Ex 13

Modifier le professeur principal de la classe "6eme B" (id_classe = 4).

Nouveau prof : id = 11

```sql
UPDATE principal
SET professeur_id = 11
WHERE classe_id = 4
```