SYSTÈME DE GESTION DE BASES DE DONNÉES

Langage d'interrogation de données (LID)

Base de données relationnelle

Une base de données relationnelle permet d'organiser les données en **tables** (appelés relations).

Chaque case de la table contient une information atomique (non décomposable).

Chaque **ligne** de la table correspond à un objet que l'on veut gérer dans la base de données : une voiture, une personne, une espèce…

Propriété et domaine (colonne)

Chaque **colonne** de la table correspond à une propriété des objets qui se trouvent dans la table ; tous les objets de la table partagent donc les mêmes propriétés.

Chaque colonne de la table est associée à un domaine de valeur fixé a priori, par exemple : entier, texte, booléen...

Langage de manipulation de données

Le langage de données est un langage informatique permettant de décrire et de manipuler les schémas et les données d'une base de données.

SQL est le langage consacré aux SGBD relationnels.

Il permet de :

créer des tables, en définissant le domaine de chaque colonne ;

insérer des lignes dans les tables;

lire les données entrées dans la base de données.

Introduction aux bases de données relationnelles.

- Interroger les bases de données relationnelles.
  - ✓ Le langage SQL : Structured Query Language.
  - LID (*SELECT*).
  - LDD (*CREATE, ALTER*).
  - LMD (*INSERT, UPDATE, DELETE*).
  - LCD (*GRANT, REVOKE,)*
  - LCT (*COMMIT, ROLLBACK*).
- Les bases de données relationnelles.
  - ✓ Nécessité de stocker les données structurées.
  - ✓ Autres méthodes de stockage des données.
  - ✓ Les SGBDR les plus répandus (*Mysql, postgreSQL, …*).
- La clé primaire.
- ✓ Qu'est-ce qu'une clé primaire ?
- ✓ Clé candidate
- ✓ Clés naturelles et clés artificielles.
- ✓ Numéro automatique.
- Clé étrangères et intégrité référentielle.
  - ✓ La clé étrangère.
  - ✓ La contrainte d'intégrité référentielle.
- ✓ Intégration des requêtes dans une application (*PHP*).
- Les requêtes imbriquées
- Nécessité de protéger les données

Le langage SQL : Structured Query Language

LID : « Langage d'Interrogation de Données » : permet de rechercher des informations utiles en interrogeant la base de données.

LDD : « Langage de Définition de Données » : permet la définition et la mise à jour de la structure de la base de données (tables, attributs, vues, index, ...).

LMD : « Langage de Manipulation de Données » : permet de manipuler les données de la base et de les mettre à jour.

LCD : « Langage de Contrôle de Données » : permet de définir les droits d'accès pour les différents utilisateurs de la base de données, donc il permet de gérer la sécurité de la base et de confirmer et d'annuler les transactions.

Langage d'interrogation des données (LID)

Le langage d'interrogation de données (LID) permet d'**établir une combinaison d'opérations** portant sur des tables (relation). Le résultat de cette combinaison d'opérations est lui-même une table temporaire.

la **PROJECTION** qui permet de garder certains attributs ou colonnes d'une table.

la **SELECTION** qui permet de conserver certaines lignes (ou tuples) d'une table en fonction d'un critère (filtre)

la **JOINTURE** qui permet de rapprocher 2 tables différentes ayant **obligatoirement** une clé commune (condition)

Données des élèves

numElv	nomElv	pnomElv	sexeElv	rueElv	villeElv	cpElv	lnElv	dnElv	codeclElv
171	PICARD	Pascal	H	4 rue du Pont	Saint-Joseph	97480	Tampon	1997-12-30	T
172	ANNETTE	Nicolas	H	18 rue du Château	Saint-Joseph	97480	Aviron	1998-01-12	S
173	PALENIK	Petra	H	23 rue du Calvaire	Saint-Joseph	97480	Saint-Pierre	1996-08-14	T
174	PEREZ MORAIS	José-Luis	H	22 rue des jardins	Saint-Joseph	97480	Saint-Pierre	1997-12-28	T
175	TOLEDE	Marlène	F	53 impasse Tourneur	Saint-Joseph	97480	Saint-Pierre	1998-01-06	S
176	RAMIN	Karoliina	F	120 boulevard Voltaire	Saint-Joseph	97480	Saint-Pierre	1996-08-01	T
177	YOUNG	Orion	H	1 rue des Indes	Saint-Joseph	97480	Saint-Pierre	1997-12-16	P
178	ABIVEN	Jule	H	14 rue du Caire	Saint-Joseph	97480	Saint-Pierre	1998-06-12	S
179	AMBROSETTI	Nicolas	H	10 rue du 8 mai 1945	Saint-Joseph	97480	Saint-Louis	1996-08-16	P

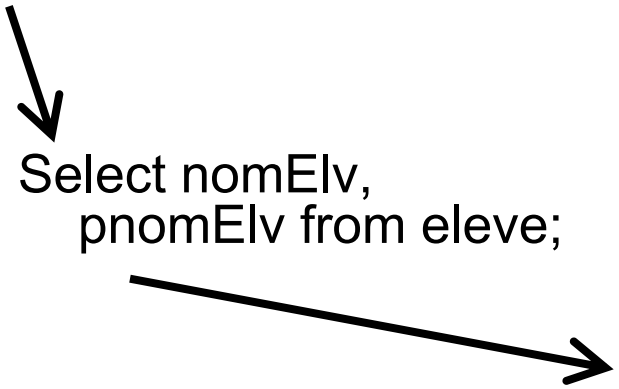
La projection

- Opération qui consiste à extraire toutes lignes d'une table pour un sous ensemble de colonnes
- Exemple :

  Liste des nom,prénom des élève

  Select nomElv, pnomElv from eleve;

nomElv	*pnomElv*	*lnElv*	*dnElv*
Eponge	Bob	Saint-Joseph	30/12/1997
Dalton	Joe	Saint-Pierre	12/10/1998
LUKE	Lucky	Saint-Pierre	11/08/1996



nomElv	*pnomElv*
Eponge	Bob
Dalton	Joe
LUKE	Lucky

La sélection

- Permet de retenir les lignes répondant à une condition de sélection
- La condition est exprimée à l'aide des opérateurs =, >, <, >=, <=, <>, opérateurs logiques de base ET, OU, NON et éventuellement des parenthèses

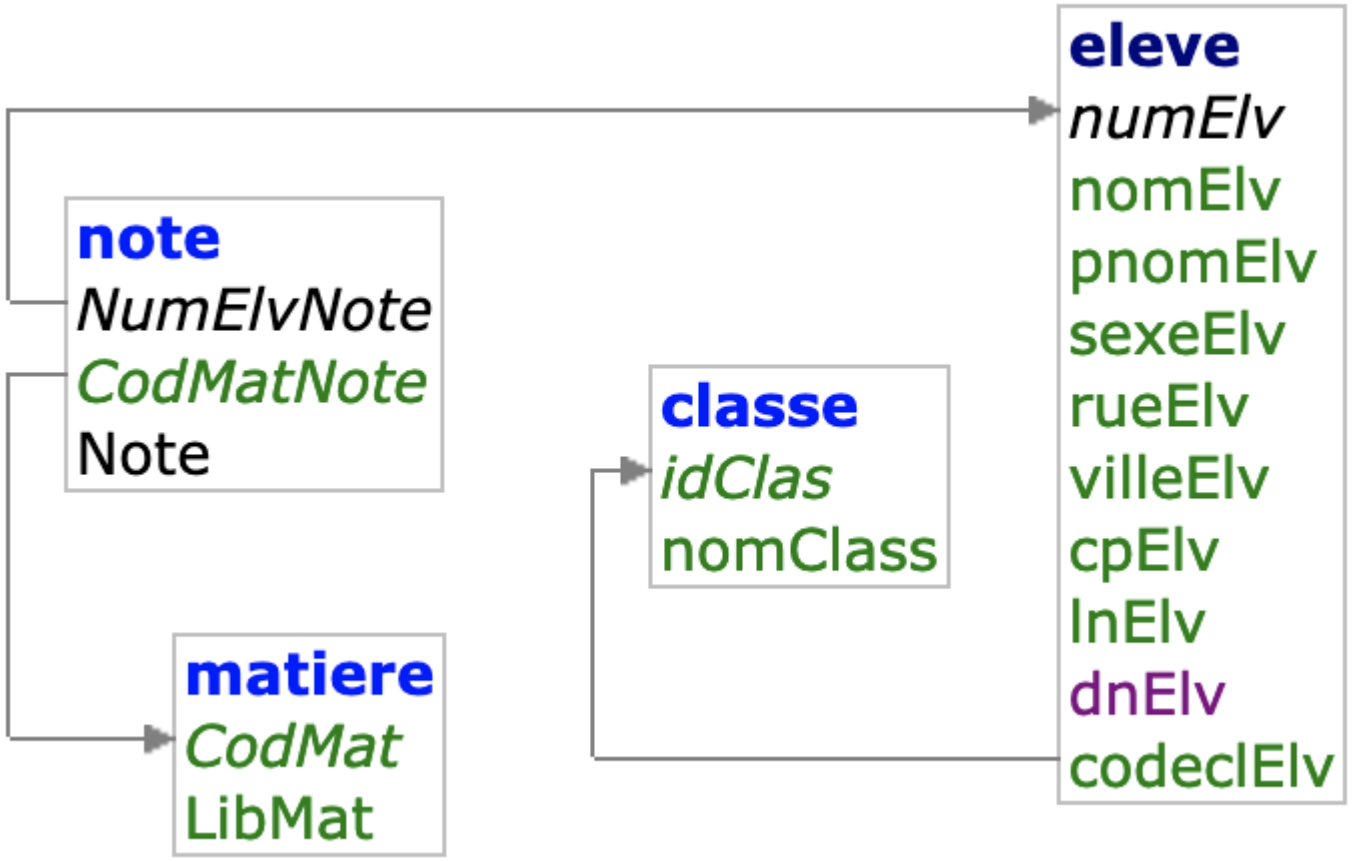
- Exemple :

nom	*prénom*	*ln*	*dn*
Eponge	Bob	Saint-Joseph	30/12/1997
Dalton	Joe	Saint-Pierre	12/10/1998
LUKE	Lucky	Saint-Pierre	11/08/1996

SELECT *nom*, *prénom*
FROM ***élève***
Where ln='Saint-Pierre' ;

nom	*prénom*
Dalton	Joe
LUKE	Lucky

Base de données des élèves



La jointure

- **Cet opérateur porte sur 2 tables qui doivent avoir au moins un attribut défini dans le même domaine**
- **La condition de jointure peut porter sur l'égalité d'un ou de plusieurs attributs définis dans le même domaine**

table élèves

nomElv	*pnomElv*	*lnElv*	*dnElv*	codeclElv
Eponge	Bob	Saint-Joseph	30/12/1997	T
Dalton	Joe	Saint-Pierre	12/10/1998	P
LUKE	Lucky	Saint-Pierre	11/08/1996	T

table classe

idClas	*nomClas*
T	Terminale
P	Première
S	Seconde

SELECT eleve.*, *nomClas* FROM eleve
INNER JOIN classe ON **eleve.codeclElv = classe.idClas** ;

Résultat de la requête SQL

nomElv	*pnomElv*	*lnElv*	*dnElv*	*codeclElv*	*nomClas*
Eponge	Bob	Saint-Joseph	30/12/1997	T	Terminale
Dalton	Joe	Saint-Pierre	12/10/1998	P	Première
LUKE	Lucky	Saint-Pierre	11/08/1996	T	Terminale

Utiliser l'ordre SELECT

SELECT [DISTINCT] * / Liste d'attributs
FROM nom(s) de table(s)
INNER JOIN jointure(s)
WHERE condition(s) de sélection
GROUP BY nom(s) de l'attribut de regroupement
HAVING condition(s) de regroupement
ORDER BY critère(s) de tri

DISTINCT

- Le résultat d'un SELECT étant un ensemble, il peut y avoir des doublons
- Le mot clé DISTINCT permet de préciser que l'on ne veut qu'un seul exemplaire de ces enregistrements
- Remarque : le résultat est une liste triée

- TAF : *liste des prénom des élèves*

SELECT pnomElv FROM eleve;
SELECT DISTINCT pnomElv FROM eleve;

Opérateurs Arithmétiques

- Vous pouvez utiliser les opérateurs arithmétiques *** / + -** pour effectuer des calculs
- Priorité des opérateurs : * et / sont prioritaires sur + et –
- Vous pouvez utiliser les parenthèses

- **TAF** : *appliquer le coefficient 2 à toutes les notes*

```
SELECT Note, 2*Note
FROM note;
```

Les Valeurs NULL

- Une valeur NULL est une valeur non disponible, non affectée, inconnue ou inapplicable. Une valeur NULL est différente du zéro ou de l'espace. Le zéro est un chiffre et l'espace est un caractère
- **TAF** :

les élèves dont nous n'avons pas le lieu de naissance

```
SELECT *
FROM eleve
WHERE lnElv IS NULL;
```

Alias de Colonne

- Renomme l'en-tête de colonne
- Est utile dans les calculs
- Suit immédiatement le nom de la colonne avec le mot-clé AS
- Doit obligatoirement être inclus entre guillemets s'il contient des espaces, des caractères spéciaux ou si il faut tenir compte des majuscules/minuscules

TAF :

Liste des note avec un coefficient de 2 avec comme entête de colonne Note coefficientée

SELECT 2*Note AS "Note coefficientée" FROM note;

Opérateur de Concaténation

- L'opérateur de concaténation (+) permet de concaténer des colonnes à d'autres colonnes, à des expressions arithmétiques ou à des valeurs constantes

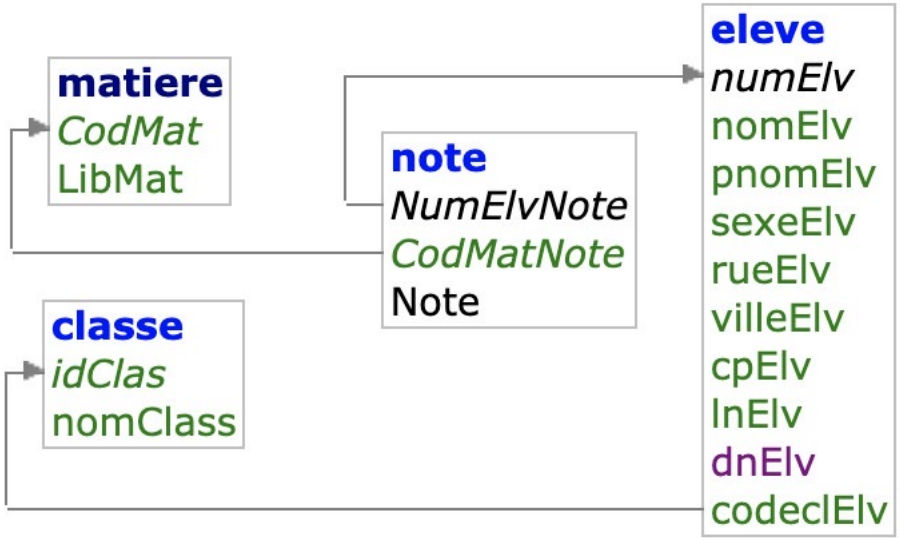
- Exemple

Liste du non et prénom des élèves dans une seule colonne (NP)

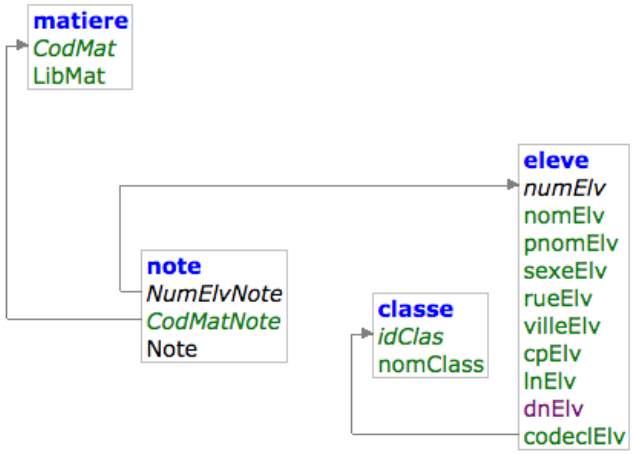
SELECT nomElv + ’ ‘ + pnomElv AS NP
FROM eleve;

MySQL :
SELECT CONCAT(nomElv, pnomElv) AS NP FROM eleve;

Schéma de la base de données des élèves



Exo01 : Utiliser SELECT

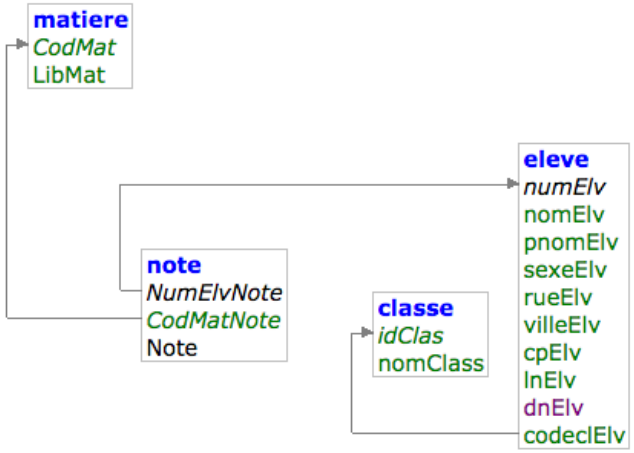


Écrire les requêtes qui permettent :

- Afficher la liste de tous les élèves
- Afficher la liste des villes de naissance sans doublons
- Afficher une liste qui comprend deux colonnes : une pour le nom et prénom concaténés et une autre pour l'adresse
- Afficher la liste des lieux et dates de naissance en définissant des alias pour ces deux colonnes

classe (idClas , nomClass)
eleve (numElv , nomElv , pnomElv , sexeElv, rueElv, villeElv, cpElv, lnElv, dnElv #codeclElv)
matiere (codMat , nomMat)
note (#numElvNote , #codMatNote , Note)

Corrigé
Exo01 : Utiliser SELECT



Écrire les requêtes qui permettent :

- Afficher la liste de tous les élèves

```
SELECT * FROM eleve ;
```

- Afficher la liste des villes de naissance sans doublons

```
SELECT DISTINCT lnElv FROM eleve;
```

- Afficher une liste qui comprend deux colonnes : une pour le nom et prénom concaténés et une autre pour l'adresse

```
SELECT DISTINCT nomElv AS "Nom", pnomElv AS "Prénom" FROM eleve;
```

- Afficher la liste des lieux et dates de naissance en définissant des alias pour ces deux colonnes

```
SELECT DISTINCT lnElv AS "Lieux de naissance", dnElv AS "Dates de naissance"
FROM eleve;
```

- Donner la liste des notes des matières M3 et M4 avec le coefficient de 3

```
SELECT Note * 3 AS "Notes M3 avec le coefficient 3"
FROM note
WHERE codMatNote = "M3" OR codMatNote = "M4";
```

Opérateur LIKE

- L'opérateur LIKE permet d'effectuer une comparaison partielle pour les chaines de caractères de type alphanumériques
- Il utilise les jokers % et _ ('pour cent' et 'souligné bas' 'underscore').
- Le joker % remplace n'importe quelle chaîne de caractères, y compris la chaîne vide. L'underscore remplace un et un seul caractère.
- Exemple :
  - Liste des élèves nommés Hoareau ou Hoarau

    SELECT * FROM élève
    WHERE nom LIKE 'Hoar%au';
  - Liste l'année de naissance des élèves Toto, Titi et Tata

    SELECT nom, dn FROM élève
    WHERE nom LIKE 'T_t_ '

SELECT <liste de colonnes>
FROM <liste de tables> + <critère de jointure>
[WHERE <critère de jointure>
AND/OR <critère de sélection>]
[GROUP BY <attributs de partitionnement>]
[HAVING <citère de restriction>]

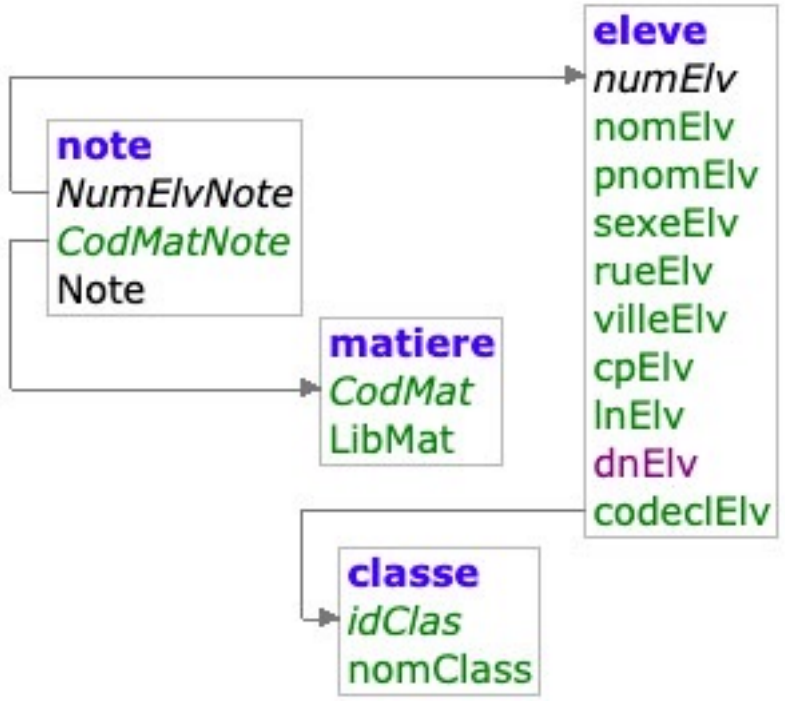
L'Opérateur BETWEEN

- L'opérateur BETWEEN permet d'afficher des lignes en fonction d'un intervalle de valeurs
- Vous spécifiez un intervalle comprenant une limite inférieure et une limite supérieure

- Exemple : *Liste des notes de la mention assez bien*

```
SELECT *
FROM note
WHERE note BETWEEN 12 AND 13,99;
```

classe (idClas , nomclass)
eleve (idElv , nomElv , prenomElv , sexeElv, rueElv, villeElv, cpElv, lnElv, dnElv #codeclElv)
matiere (idMat , libMat)
note (#idElvNote , #idMatNote , note)



Application

Liste des élève nés en 1999 et 2000

SELECT *
FROM eleve
WHERE dnElv BETWEEN *01/01/1999* **AND** *31/12/2000*;

Opérateur IN

- Permet de comparer une expression avec une liste de valeurs, utilisez l'opérateur IN

- Opérateur d'appartenance à un ensemble de valeurs

- Exemple :*Liste des élèves nés au Tampon, aux Avirons ou à Saint-Louis*

```
SELECT *
FROM élève
WHERE lnElv IN ('Tampon','Saint-Louis','Avirons') ;
```

classe (idClas , nomclass)
eleve (idElv , nomElv , prenomElv , sexeElv, rueElv, villeElv, cpElv, lnElv, dnElv #codeclElv)
matiere (idMat , libMat)
note (#idElvNote , #idMatNote , note)

Application

Liste des élèves qui ne sont pas nés au Tampon, aux Avirons ou à Saint-Louis

```
SELECT *
FROM élève
WHERE ln NOT IN ('Tampon','Saint-Louis','Avirons') ;
```

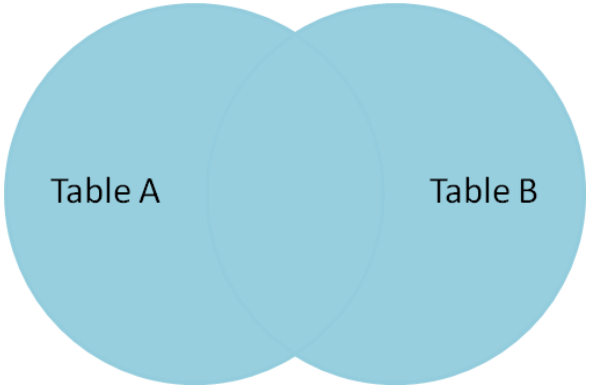
Opérateurs Logiques

- Un opérateur logique combine le résultat de deux conditions pour produire un résultat unique ou inverse le résultat d'une condition unique.
- SQL inclut trois opérateurs logiques :

AND Intersection
OR Union
NOT négation

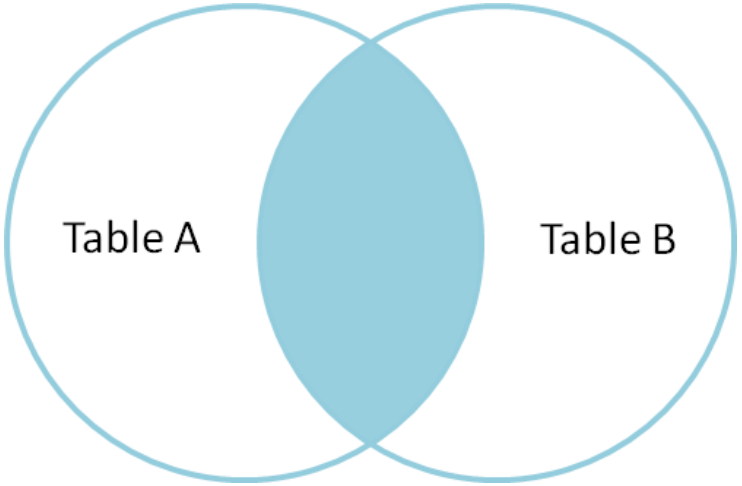
Union (OU)

- Cet opérateur porte sur deux tables qui doivent avoir le même nombre d'attributs définis dans le même domaine. On parle de tables ayant le même schéma.
- La table résultat possède les attributs des tables d'origine et les lignes de chacune, avec ou sans élimination des doublons éventuels.



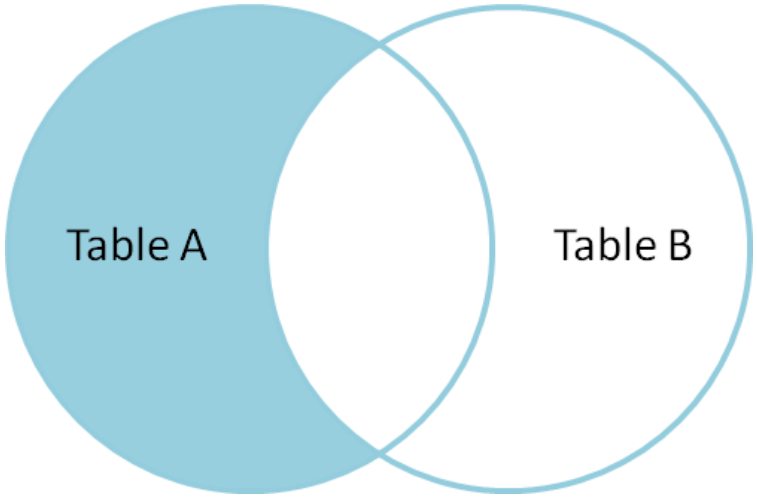
Intersection (ET)

- Cet opérateur porte sur deux tables de même schéma.
- La table résultat possède les attributs des tables d'origine et les lignes communes à chacune.



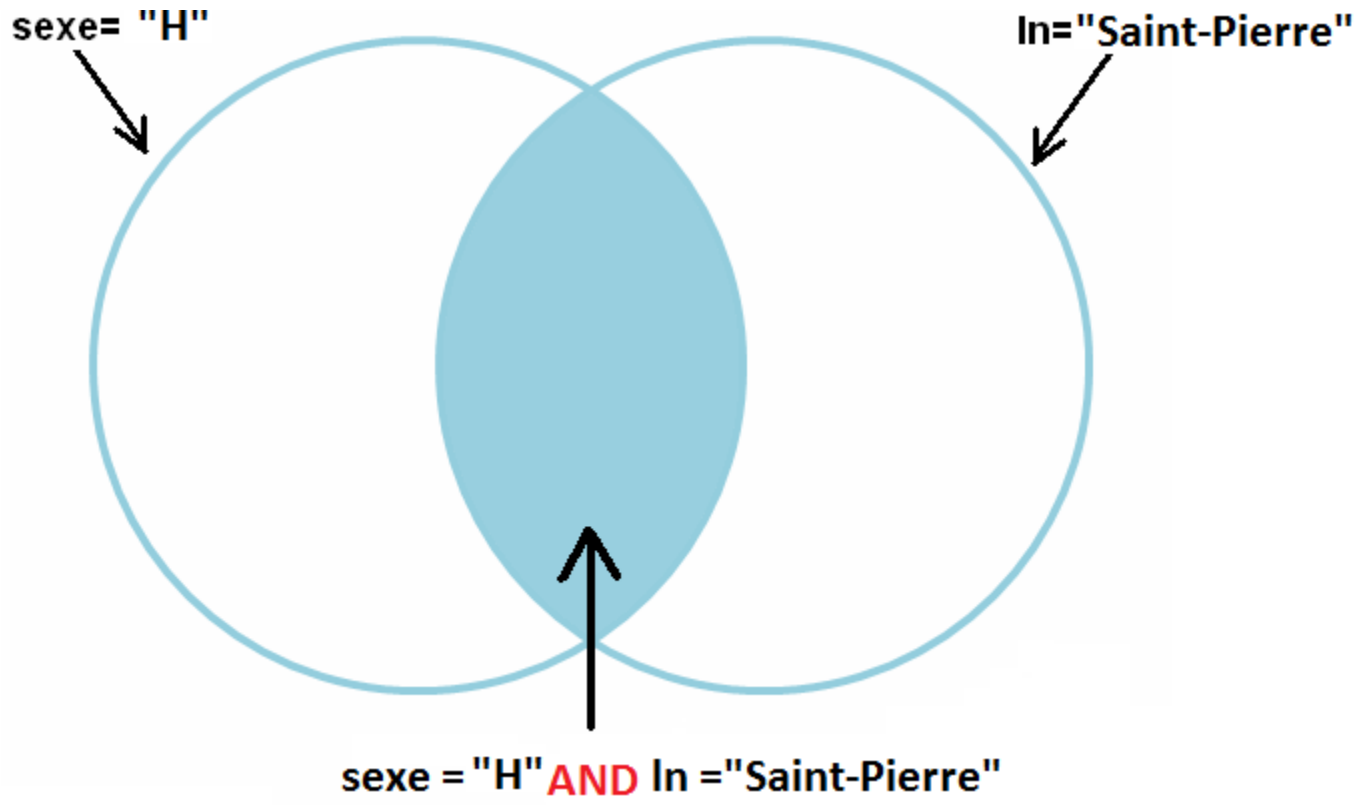
Différence (NOT)

- Cet opérateur porte sur deux tables de même schéma
- La table résultat contient les lignes de la première table qui n'appartiennent pas à la deuxième



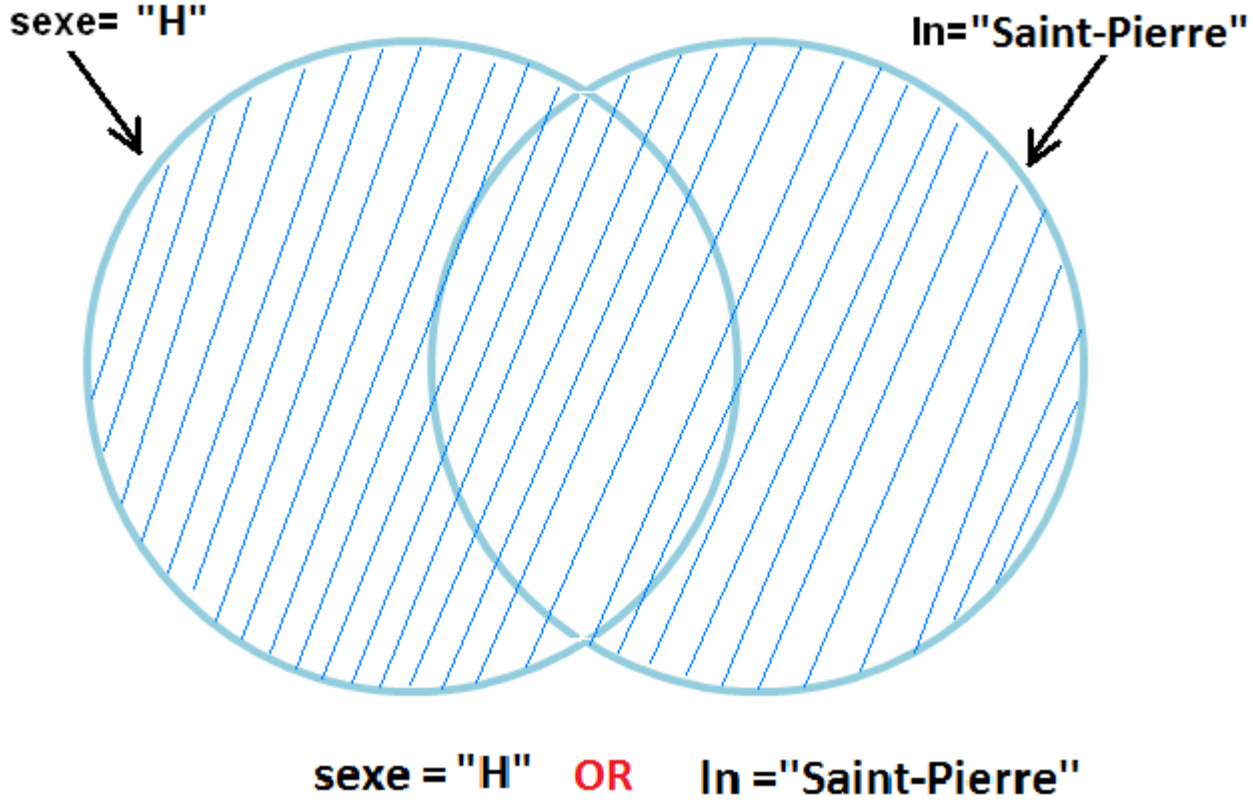
L'opérateur AND = intersection

sexe='H' **AND** ln='Saint-Pierre'



L'opérateur OR = Union

sexe = 'H' **OR** ln='Saint-Pierre'



Le tri ORDER BY

- ORDER BY colonne1 | 1 [ASC ou DESC] [, colonne2 | 2 [ASC ou DESC] ...
- La clause ORDER BY sert à trier les lignes
- **ASC (ascending)** classe les lignes en ordre croissant. C'est l'ordre par défaut.
- **DESC (descending)** classe les lignes en ordre décroissant.

Exemple : *Liste des élèves triée par sexe et note.*

```
SELECT * FROM élève
ORDER BY sexe , note DESC;
```

Syntaxe générale SQL LID

SELECT [DISTINCT] < noms des champs / * >
FROM < noms des tables et jointure(s) >
[**WHERE** < conditions de sélection et jointure(s)> AND / OR]
[**GROUP BY** < noms des champs >]
[**HAVING** < conditions de having >]
[**ORDER BY** < noms des champs > [ASC / DESC]
;

La clause TOP ou LIMIT

- La clause TOP n permet d'afficher les n premières lignes de la requête

Exemple : *liste des 5 meilleures notes*

```
SELECT *
FROM note
ORDER BY note DESC
LIMIT 5;
```

```
SELECT TOP 5 *
FROM note
ORDER BY note DESC;
```

Les Jointures

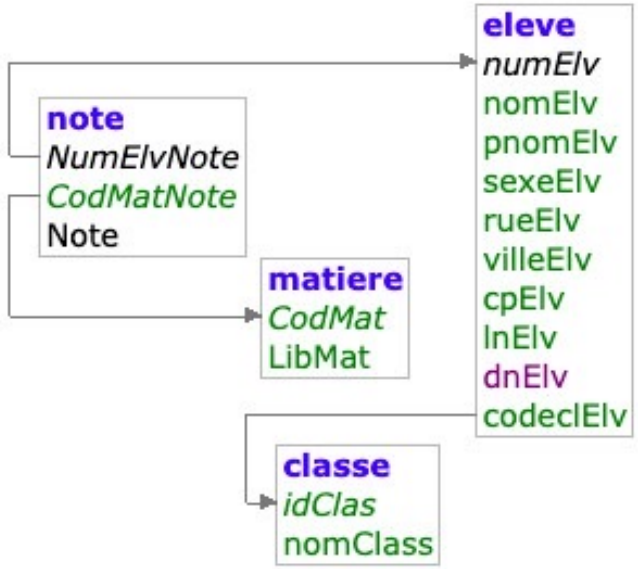
Les Jointures

Les jointures en SQL permettent d’associer plusieurs tables dans une même requête.

Cela permet d’obtenir des résultats qui combinent les données de plusieurs tables.

On exploite ici la puissance des bases de données relationnelles.

Jointures

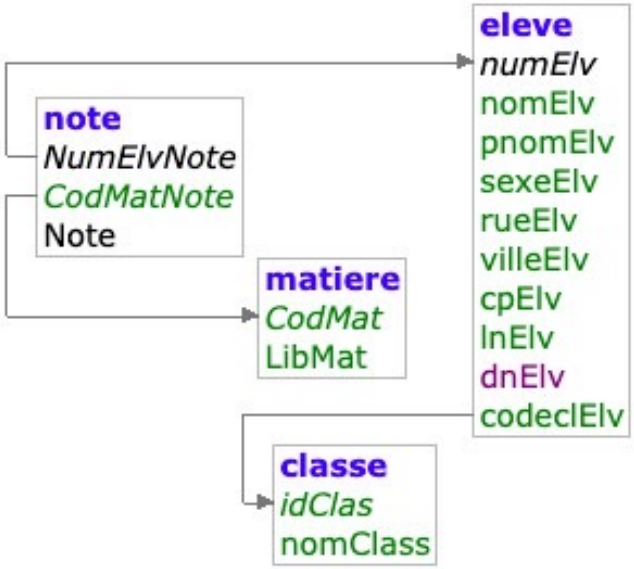



```
SELECT ...
FROM  nom_table1, ...
INNER JOIN nom_table2 ON critère;
INNER JOIN ...
```

- pas de condition de sélection : résultat obtenu = produit cartésien des tables présentes derrière le FROM.
- Requête : lister des notes d'un élève?

SELECT nomElv, pnomElv, note
FROM eleve
INNER JOIN note **ON** eleve.numElv = note.numElvNote

Liste des matières par élève

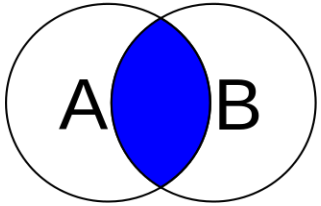



```
SELECT e.nomElv
   , e.pnomElv
   , m.LibMat
FROM eleve e
INNER JOIN note n
   on e.numElv = n.NumElvNote
INNER JOIN matiere m
   on m.CodMat = n.CodMatNote
ORDER BY e.nomElv ASC;
```

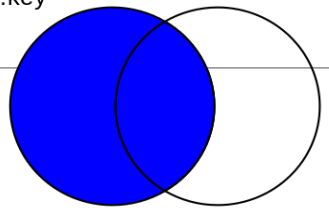
```
SELECT e.nomElv
   , e.pnomElv
   , m.LibMat
FROM eleve e, note n, matiere m
WHERE e.numElv = n.NumElvNote
AND m.CodMat = n.CodMatNote
ORDER BY e.nomElv ASC;
```

SQL JOINS

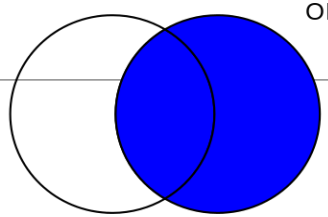
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



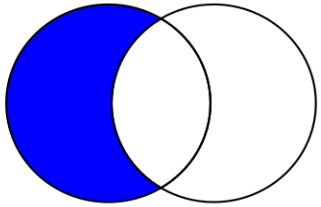
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



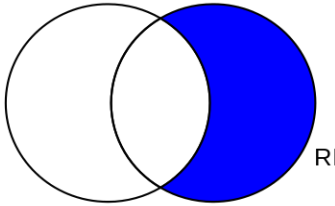
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



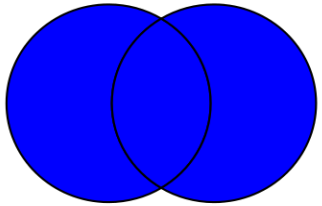
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



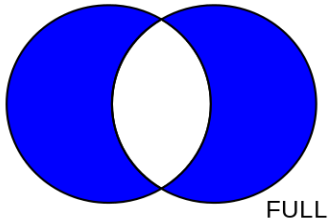
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



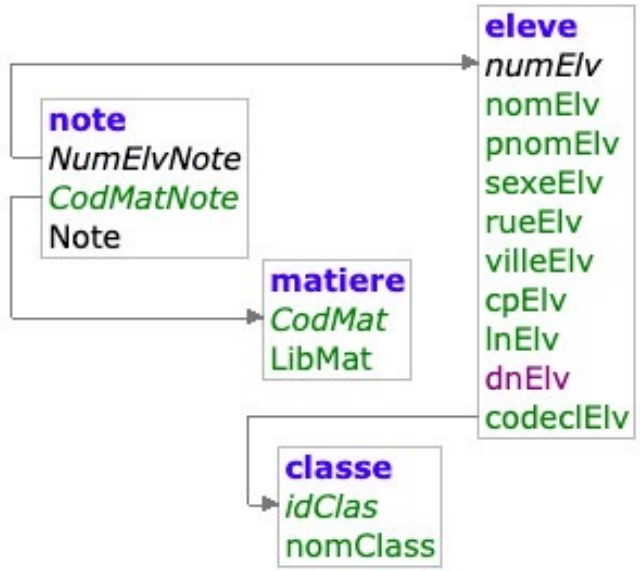
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL



Syntaxe générale SQL LID

SELECT [DISTINCT] < noms des colonnes / * toutes le colonne >
FROM < noms des tables et jointure(s) >
[**WHERE** < conditions de sélection et jointure(s)>]
[**GROUP BY** < noms des champs >]
[**HAVING** < conditions de group by >]
[**ORDER BY** < noms des champs > [ASC / DESC \]
LIMIT nombreLignes **OFFSET** premiereLigne
;

Exo02 :
la clause WHERE



Développer les requêtes suivantes :

- Liste des élèves ayant une note supérieure à 12
- Liste des élèves de sexe féminin (F)
- Liste des élèves nés à Saint-Louis puis ceux qui ne sont pas nés à Saint-Pierre
- Liste des élèves qui portent les prénoms : Joe, Jack ou Avrell
- Les élèves qui habitent Saint-Pierre
- Les élèves de sexe masculin ayant une de note de plus de 12
- Les élèves ayant une note entre 12 et 14
- Liste des élèves triée dans l'ordre décroissant des notes
- Liste des élèves triée dans l'ordre croissant par sexe puis décroissant par nom

classe (idclas , nomclass)
eleve (idElv , nomElv , prenomElv , sexeElv, rueElv, villeElv, cpElv, lnElv, dnElv #codeclElv)
matiere (codMat , Lbat , coefMat)
note (#numElvNote , #CodMatNote , Note)

Les Fonctions de date

Fonctions Date

Format de la Date :
Année, mois, jour, heure, minute, seconde, milliseconde

Les différents types de dates que peut stocker MySQL :

DATE : stocke une date au format AAAA-MM-JJ (Année-Mois-Jour) ;

TIME : stocke un moment au format HH:MM:SS (Heures:Minutes:Secondes) ;

DATETIME :
stocke la combinaison d'une date et d'un moment de la journée au format AAAA-MM-JJ HH:MM:SS.

TIMESTAMP : stocke une date et un moment sous le format AAAAMMJJHHMMSS ;

YEAR : stocke une année, soit au format AA, soit au format AAAA.

Il faut surtout retenir
DATE (AAAA-MM-JJ) quand le moment de la journée n'est pas necessaire.
DATETIME (AAAA-MM-JJ HH:MM:SS) si vous avez besoin du jour et de l'heure précise à la seconde près.

SELECT nom, date FROM eleve WHERE dn = '1998-04-02'

DAY, MONTH et YEAR

- Ces fonctions renvoient respectivement le jour, mois et année d'une date donnée ou la date du jour

- Syntaxes :
  - DAY **(** *date* **)**
  - MONTH **(** *date* **)**
  - YEAR **(***date***)**

  Date du jour :
  - CURDATE()
  - NOW()

Fonctions Date

Les fonctions scalaires permettant la manipulation de nombres, des dates ou des chaînes de caractères, etc

Exemple Liste des élèves nés un Jeudi (*Thursday*)

```
SELECT nomElv, DAY(dnElv) AS jour FROM eleve
order by jour;
```

```
SELECT nom, DAY(dn) AS jour, MONTH(dn) AS mois, YEAR(dn) AS annee,
HOUR(dn) AS heure, MINUTE(dn) AS minute, SECOND(dn) AS seconde
FROM eleve
```

La fonction DATEDIFF

DATEDIFF : renvoie le nombre de limites de date et d'heure traversées entre deux dates données.

Syntaxe
DATEDIFF **(** *intervalle* **,** *date1*, *date2* **)**
Syntaxe MySQL
DATEDIFF **(***date1*, *date2* **)**
TIMESTAMPDIFF(unité, date1, date2)

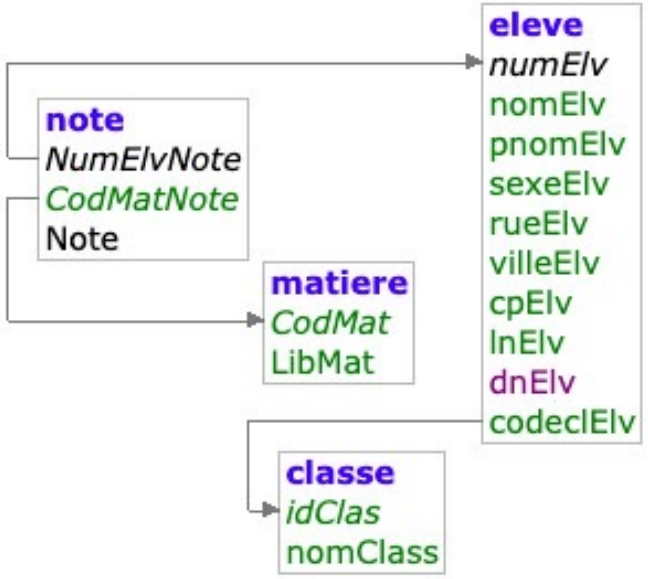
Unité :
SECOND (secondes), MINUTE (minutes), HOUR (heures), DAY (jours), WEEK (semaines), MONTH (mois), QUARTER (trimestres) et YEAR (années).

Les Fonctions SQL
utiles pour les dates et les heures

Exo03: gestion des dates

Développer les requêtes suivantes :

- Liste des noms et année de naissance des élèves
- Liste des élèves nées un 30
- Liste des élèves nés au mois de décembre
- Les noms et prénoms des élèves nés entre 15/3/1997 et le 14/3/1998
- Liste des noms et âges des élèves



Exo03: gestion des dates *corrigé*

Développer les requêtes suivantes :

1. Liste des noms et année de naissance des élèves
2. Liste des élèves nées un 30
3. Liste des élèves nés au mois de décembre
4. Les noms et prénoms des élèves nés entre 15/3/1997 et le 14/3/1998
5. Liste des noms et âges des élève (âge sur l'année civile) 5bis (âge à la date du jour)
6. Liste des élèves nés un Thursday (Jeudi)

1. SELECT nomElv, year(dnElv) FROM eleve;
2. SELECT nomElv, DAY(dnElv) as jour FROM eleve WHERE DAY(dnElv) = 30;
3. SELECT nomElv, MONTH(dnElv) as mois FROM eleve WHERE MONTH(dnElv) = 12 ORDER BY nomElv;
4. SELECT nomElv,pnomElv, dnElv FROM eleve WHERE dnElv BETWEEN '1997-03-15' AND '1998-03-14'
5. SELECT nomElv, year(curdate())-year(dnElv) as age FROM eleve

5 bis SELECT nomElv, TIMESTAMPDIFF(YEAR, dnElv, CURDATE()) AS age FROM eleve;

6 SELECT nomElv,pnomElv, dnElv FROM eleve WHERE DAYNAME(dnElv) like 'Thursday'

Autres Fonctions

Fonctions d'agrégation

FONCTION D'AGREGATION	DESCRIPTION
AVG([DISTINCT|ALL]*n*)	Valeur moyenne de *n*, en ignorant les valeurs NULL (AVERAGE)
COUNT({*|[DISTINCT|ALL]expr})	Nombre de lignes, où expr est différent de NULL. Le caractère * comptabilise toutes les lignes sélectionnées y compris les doublons et les lignes NULL
MAX([DISTINCT|ALL]*expr*)	Valeur maximale de *expr*, en ignorant les valeurs NULL
MIN([DISTINCT|ALL]*expr*)	Valeur minimale de *expr*, en ignorant les valeurs NULL
SUM([DISTINCT|ALL]*n*)	Somme des valeurs de *n*, en ignorant les valeurs NULL
MOD([DISTINCT|ALL]*n*)	Modulo d'un nombre (le reste d'une division)
Autres fonctions mathématiques	

Les fonctions

ROUND() arrondir la valeur
UPPER() afficher une chaîne en majuscule
LOWER() afficher une chaîne en minuscule
NOW() date et heure actuelle
RAND() retourner un nombre aléatoire
CONCAT() concaténer des chaînes de caractères
CURRENT_DATE() date actuelle

Les fonctions **mathématiques**

ABS() retourner la valeur absolue d'un nombre [MySQL, PostgreSQL, SQL Server]
ACOS() retourne l'arc cosinus [MySQL, SQL Server]
ASIN() retourne l'arc sinus [MySQL, SQL Server]
ATAN2() ou ATAN() retourne la tangente de 2 arguments [MySQL]
ATAN() retourne l'arc tangente [MySQL, SQL Server]
ATN2() retourne l'angle en radian entre un axe et un rayon [SQL Server]
CBRT() retourne la racine carrée de l'argument [PostgreSQL]
CEIL() obtenir la valeur entière supérieure d'un nombre [MySQL, PostgreSQL]
CEILING() obtenir la valeur entière supérieure d'un nombre [MySQL, SQL Server]
CONV() convertir des nombres entre différentes bases de nombre [MySQL]
COS() obtenir le cosinus d'une valeur numérique [MySQL, SQL Server]
COT() retourne la cotangante [MySQL, SQL Server]
CRC32() calcule une valeur de contrôle de redondance cyclique [MySQL]
DEGREES() convertir un angle en radians en degrés [MySQL, PostgreSQL, SQL Server]

https://sql.sh/fonctions/mathematiques

Alias de table

- Il est possible d'utiliser des alias des tables utilisé dans une jointure pour simplifier l'écriture de la requête SQL
-
- Exemple : *Liste des élèves par classe*

```
SELECT nomElv, nomclass as classe
FROM eleve e, classe c
WHERE e.codeclElv = c.idclas
ORDER BY  classe;
```

Regroupement

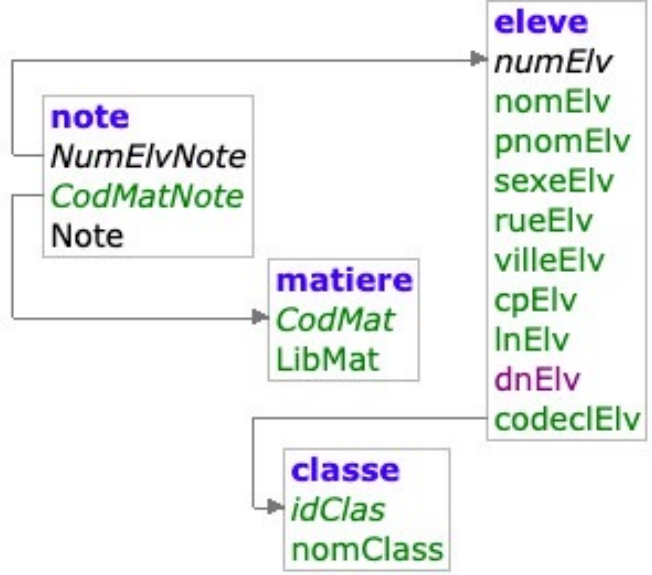

10,000
1,000
100
10
1
100,000
1,000,000


1,000,000
100,000
10,000
1,000
100
10
1

Regrouper les lignes d'une table

- Il peut être intéressant de regrouper des lignes afin de faire des opérations par groupe (opérations statistiques par exemple)
- Cette opération se réalise à l'aide de la clause *GROUP BY*, suivie du nom de chaque colonne sur laquelle on veut effectuer des regroupements.

Clause GROUP BY



- Cette clause permet de créer des sous-ensembles de lignes pour lesquels la valeur d'une (ou plusieurs) colonne est identique
- Exemple : *calculer la moyenne des notes pour chaque classe*

```
SELECT nomclass as classe, ROUND(avg(Note), 2) as Moyenne
FROM eleve e, classe c, note n
WHERE e.codeclElv = c.idclas
AND e.numElv = n.numElvnote
GROUP BY nomclass;
```

```
SELECT nomclass as classe, avg(Note) AS Moyenne
FROM eleve e, classe c, note n
WHERE e.codeclElv = c.idclas
AND e.numElv = n.NumElvnote
GROUP BY nomclass;
```

La clause HAVING

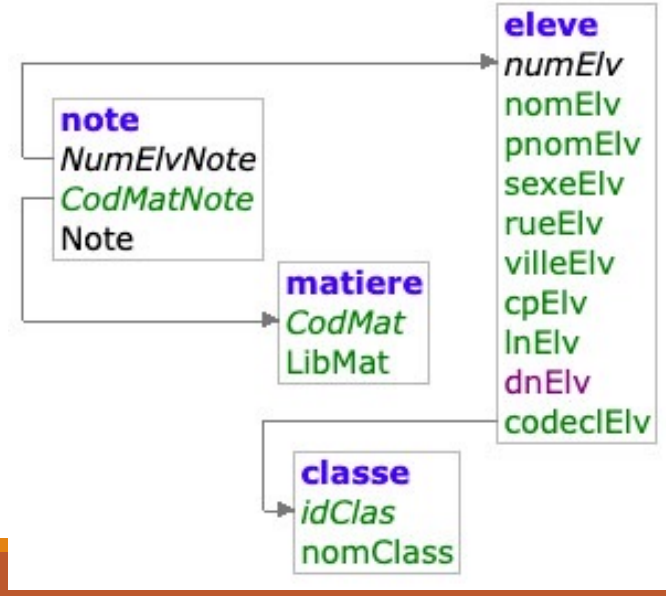
- Cette clause, contrairement à la clause WHERE qui précise les conditions à appliquer sur les lignes d'une table, permet de préciser des conditions au niveau des sous-ensembles créés par GROUP BY
- Exemple : *reprendre la requête précédente mais afficher uniquement les classes dont la moyenne est supérieur à 11*

```
SELECT nomclass as classe, ROUND(avg(Note), 2) as Moyenne
FROM eleve e, classe c, note n
WHERE e.codeclElv = c.idclas
AND e.numElv = n.numElvnote
GROUP BY nomclass
HAVING avg(Note) > 11;
```

La clause HAVING suite

- Exemple : *modifier la requête précédente mais afficher uniquement pour les classes de première et seconde celle dont la moyenne est supérieur à 10*

```
SELECT nomclass as classe, ROUND(avg(Note), 2) as Moyenne
FROM eleve e, classe c, note n
WHERE e.codeclElv = c.idclas
AND e.numElv = n.numElvnote
AND  nomclass not in ('Terminale')
GROUP BY nomclass
HAVING avg(Note) > 10;
```



Règle de regroupement

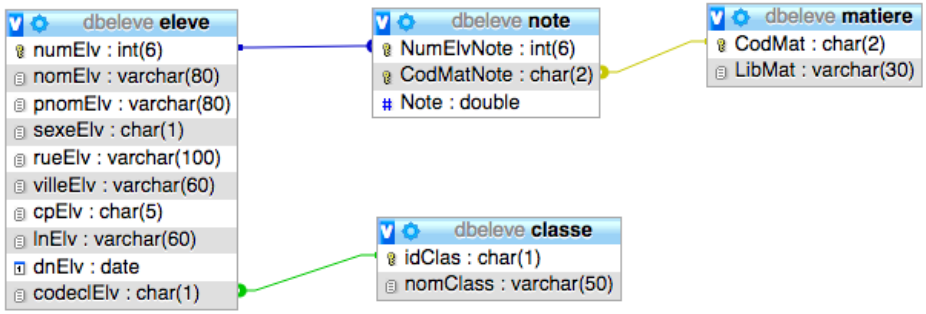
- Toutes les colonnes qui apparaissent après la clause SELECT, sujette à un regroupement, doivent :
  - Soit faire partie d'une fonction d'agrégation
  - Soit apparaître dans la clause GROUP BY

SELECT champs à afficher
FROM tables nécessaires, Jointure
WHERE condition
AND critère de jointure
GROUP BY champ sur lequel s'effectue le regroupement
HAVING condition spécifique au regroupement
ORDER BY nom du champ sur lequel s'effectue le tri
;

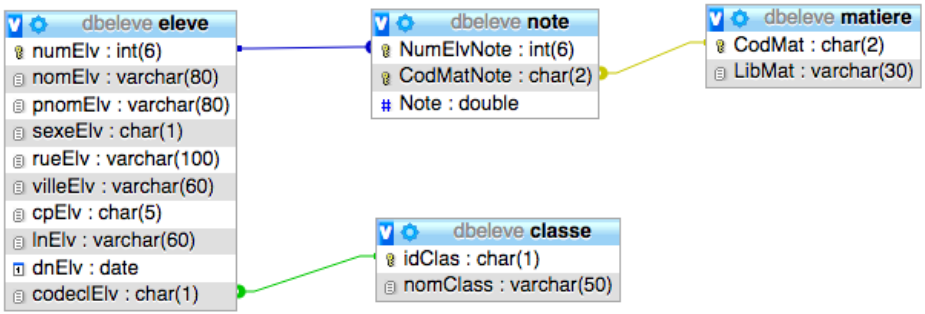
Exo04 : regrouper des données

Développer les requêtes suivantes :

- Nombre total d'élèves
- Moyenne des notes par matière en affichant le nom de la matière et la moyenne
- Nombre d'élèves dans chaque classe
- Note maximale et note minimale pour chaque matière
- Note maximale et note minimale pour chaque matière dont la note minimale est supérieur à 8 classée par note minimale par ordre décroissant,
- Moyenne des élèves de sexe féminin,
- Combien y'a-t-il de lieux de naissance distincts
- Combien y'a-t-il de prénoms différents par classe



Exo04 :
regrouper des données



Nombre total d'élèves

```
SELECT count( pnomElv)
FROM  eleve ;
```

Moyenne des notes par matière en affichant le nom de la matière et la moyenne

```
SELECT  round(avg(Note),2), libMat
FROM note n,matiere m
WHERE m.CodMat = n.CodMatNote
GROUP BY libMat;
```

Nombre d'élèves dans chaque classe

```
SELECT  nomClass as Classe , count(nomElv) as Nombre
FROM  eleve e, classe c
WHERE c.idClas = e.codeclElv
GROUP BY nomClass;
```

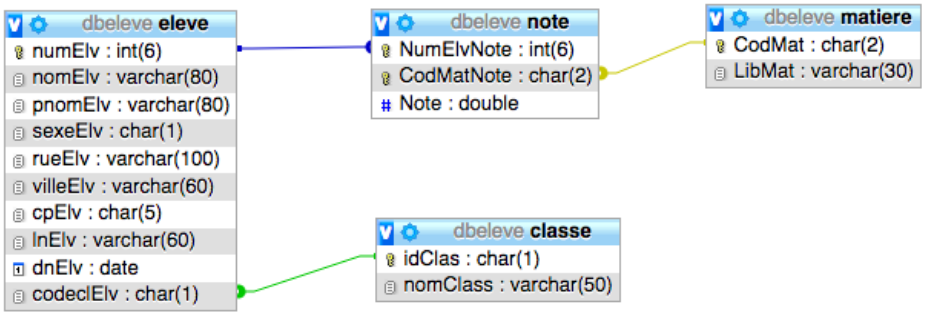
Exo04 : regrouper des données

Note maximale et note minimale pour chaque matière

```
SELECT Min(Note) as Minimum ,Max(Note) As Maximum, libMat  As Matière
FROM note n,matiere m
WHERE m.CodMat = n.CodMatNote
GROUP BY libMat;
```

Note maximale et note minimale pour chaque matière dont la note minimale est supérieur à 8 classée par note minimale par ordre décroissant,

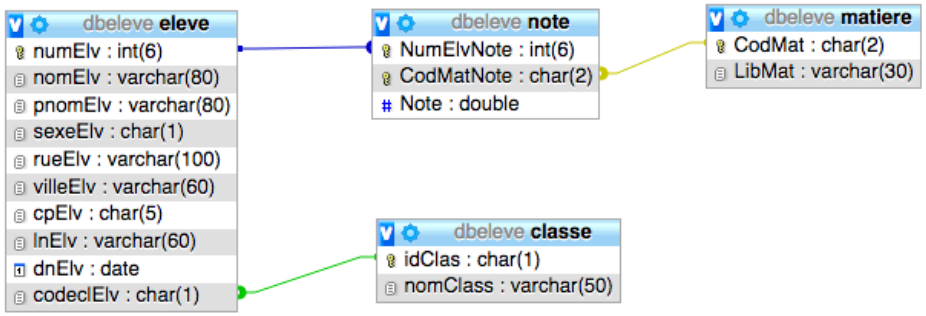
```
SELECT  Min(Note) as Minimum ,Max(Note) As Maximum, libMat
 As Matière
FROM  note n,matiere m
WHERE m.CodMat = n.CodMatNote
GROUP BY libMat
Having Max(Note)>8
Order by Min(Note)  DESC;
```



Moyenne des élèves de sexe féminin,

```
SELECT ROUND(AVG(Note),2) AS "Moyenne de filles"
FROM eleve e, note n
WHERE n.numElvNote = e. numElv
AND sexeElv = "F";
```

Exo04 :
regrouper des données



- Combien y'a-t-il de lieux de naissance distincts

```
SELECT count(distinct lnElv)
FROM eleve
```

- Combien y'a-t-il de prénoms différents par classe

```
SELECT count(distinct pnomElv), nomClass
FROM eleve, classe
WHERE  idClas = codeclElv
GROUP BY nomClass ;
```

Les opérateurs de comparaison

Les **comparateurs de chaîne** : LIKE, BETWEEN et IN sont utilisés dans une condition.

LIKE

Nous avons vu comment fonctionne l'opérateur LIKE et que cet opérateur est insensible à la casse. On l'utilise souvent avec les caractères génériques % et _

BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données.

Exemple entre tel âge et tel âge ou entre deux dates.

Age **BETWEEN** 15 **AND** 20

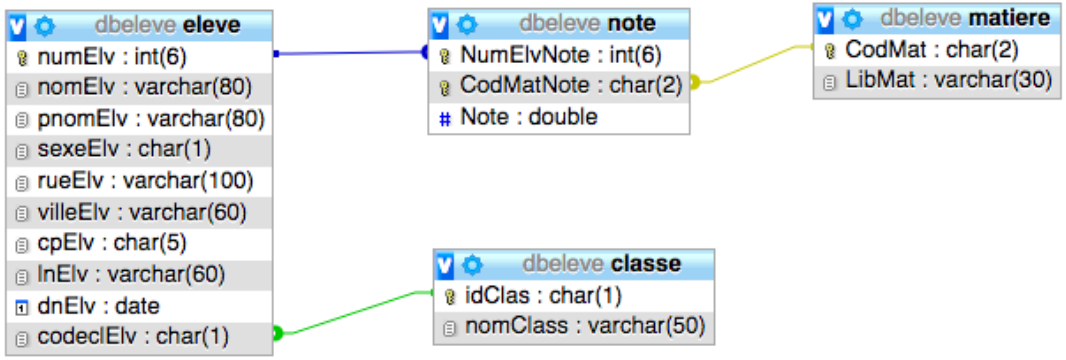
IN

L'opérateur logique IN s'utilise pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés.

Exemple les personnes .qui ont 10, 20 ou 30 ans

Age **IN** (10, 20, 30,)

Exo05 IN et BETWEEN



Développer les requêtes suivantes :

- Combiens, y a-t-il des notes pour la matière M1 comprises entre 12 et 16

Nb notes
4

- Liste des étudiants nés en Mars (03) Mai(05), Juin(06) et décembre.
- Quelle est la moyenne pour les notes comprise entre 12 et 16 dans les matières M1, M3 et M5

Matière	Moyenne
ANGLAIS	13.09
ECO-GESTION	10.44
HIST-GEO	9.70

Exo05 IN et BETWEEN

Combiens, y a-t-il de notes pour la matière M1 comprise entre 12 et 16

```
SELECT COUNT(note) "Nombre de notes"
FROM note
WHERE CodMatNote LIKE "M1"
AND Note BETWEEN 12 AND 16
```

Liste des étudiants nés en Mars (03) Mai(05), Juin(06) et décembre.

```
SELECT nomElv AS Nom, pnomElv AS Prénom
FROM eleve
WHERE MONTH(dnElv) IN (3,5,6,12);
```

Exo05 IN et BETWEEN

Quelle est la moyenne pour les notes comprise entre 12 et 16 dans les matières M1, M3 et M5

```
SELECT LibMat  "Matière", ROUND(AVG(note),2) Moyenne
FROM note
INNER JOIN matiere ON CodMat = CodMatNote
WHERE CodMatNote IN ("M1", "M3","M5")
GROUP BY CodMatNote
```

et seulement pour les matières dont la moyenne > 10

```
HAVING avg(note) > 10;
```

Quelle est la moyenne pour les notes comprise entre 12 et 16 pour toutes les matières à l'exception des matières M1, M3 et M5

```
WHERE CodMatNote NOT IN ("M1", "M3","M5")
```

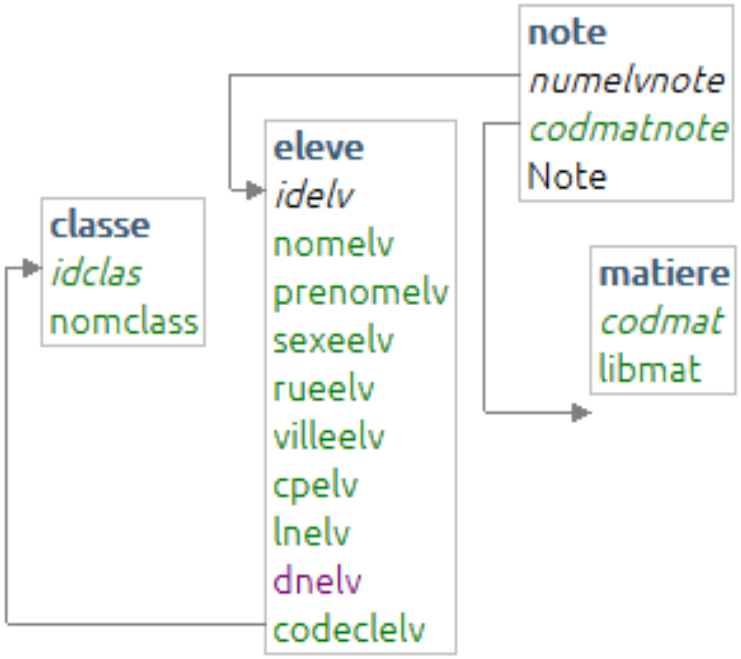
Sous requêtes

- Une sous requête est une requête SELECT qui renvoie une valeur unique et est imbriquée dans une autre instruction SELECT
- Les instructions contenant une sous-requête se présentent généralement sous une des formes suivantes :
  - **WHERE *expression* [NOT] IN (*sous requête*)**

Sous requêtes et l'opérateur IN

Exemple :

Liste des élèves n'ayant pas de note dans la matière 2 (M2 Mathématiques)



Résolution de la requête

Etape 1 lister les élève

```
SELECT nomElv as Nom, prenomElv as Prénom
FROM eleve
```

Etape 2 lister les élèves ayant une note en M2

```
SELECT e.idElv FROM eleve e,  note n
WHERE  e.idElv = n.numElvnote AND codmatnote LIKE 'M2'
```

Etape 3 mixer les deux requêtes

```
SELECT nomElv as Nom, prenomElv as Prénom
FROM eleve NOT IN (SELECT e.idElv FROM eleve e,  note n WHERE  e.idElv =
n.numElvnote AND codmatnote LIKE 'M2') ;
```

Exo06 sous requêtes

Développer les requêtes suivantes :

- Liste des élèves ayant eu la meilleur note dans la matière 3
- Liste des élèves ayant eu la dernière note dans la matière 3
- Combien y a-t-il de notes supérieures à la moyenne des notes