Bases de données
Cours I

Lycée Thiers

Une base de donnée qu'est que c'est ?

• L'utilisation de fichiers pour stocker des données est trop limiter pour gérer un grand volume de données. On utilise plutôt une base de données qui manipulera des fichiers mais organisera les données de façon à les gérer de façon optimale.

• Un bon exemple de base de données est un carnet d'adresse. Dans un carnet d'adresse chaque entrée a un prénom, nom, numéro de téléphone, adresse, etc... Toutes ces données sont en relation : une adresse est reliée à un nom-prénom, etc...

• La recherche dans un carnet d'adresse se fait en général par le nom. Mais par exemple un commercial pourra rechercher plutôt par adresse afin de gérer ses déplacements ; une centrale d'appel par numéro de téléphone pour optimiser ses coûts de frais d'appel ; une personne dont le véhicule est en panne pourra y rechercher plutôt un garagiste, etc...

• Une base de donnée répond à des requêtes formulées dans un langage structuré de requêtes : ce langage permettra de créer/modifier ou d'accéder aux informations pertinentes à l'aide d'un langage approprié. Par exemple : "Quels sont les garagistes dans mon carnet d'adresse situés sur la commune de mon lieu de panne ?".

Base de donnée - Vocabulaire

• Illustrons ces propos en imaginant quelle pourrait être une base de donnée pour gérer tous ses contacts.

• On pourrait utiliser un carnet d'adresse pour stocker toutes les informations concernant ses amis : prénom, nom, numéro(s) de téléphone (1 ou plusieurs), adresse, date anniversaire.

• Et un autre carnet d'adresse pour stocker toutes les informations concernant ses contacts professionnels : prénom, nom, numéros(s) de téléphone, adresse, société, position.

• Ces deux carnets d'adresse constitueront deux tables dans la base de donnée ; la table est définie par son nom et son schéma de relation : c'est l'ensemble des champs (attributs) qui la constituent (nom, prénom, téléphone, etc...).

• Les deux tables et leurs schémas relationnels :

Amis				
Prénom	Nom	Téléphone	Adresse	Anniversaire

Contacts Pro.					
Prénom	Nom	Téléphone	Adresse	Société	Poste

Base de donnée - Vocabulaire

• Chaque table va regrouper des contacts, *n*-uplets de valeurs. Ils ont appelés enregistrement.
Par exemple on a ajoute un enregistrement. :

Amis				
Prénom	Nom	Téléphone	Adresse	Anniversaire
Paul	Dupond	0102030405	13 av. de la lumière	27 mars 1990

• Cet enregistrement est constitué des valeurs d'attributs : Paul, Dupond, 0102030405, etc...

• Chaque attribut a un domaine ; c'est essentiellement son type : prénom, nom et adresse seront stockés comme chaîne de caractère ; téléphone comme un entier et anniversaire comme une date.

Base de donnée - Vocabulaire

• Dans une table on doit pouvoir accéder facilement à un enregistrement à l'aide de la valeur de l'un de ses attributs.

• Un ou plusieurs attributs donnant accès à un unique élément dans une table s'appelle une clé.

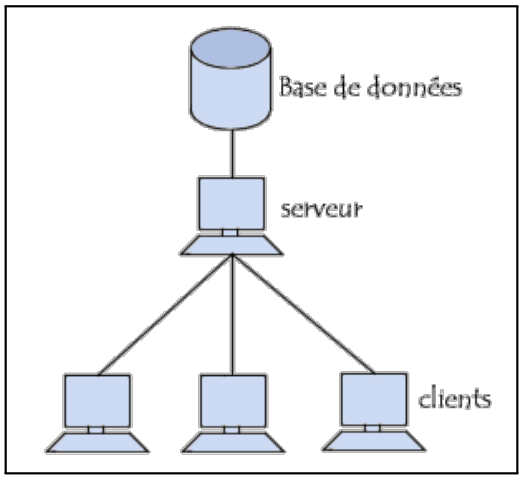
• Par exemple le couple (nom, prénom) est un assez bon choix de clé. On parle de clé composite quand elle est constituée de plusieurs attributs.

• La clé choisie s'appelle la clé primaire. Elle doit être aussi simple que possible.

• Les autres clés possibles sont les clés secondaires.

Base de données. Architecture clients-serveur

• Une base de donnée est le système permettant le stockage des données, de manière structurée, en général sur un serveur (ordinateur relié à un réseau, internet par exemple). Elle est conçue selon une architecture clients-serveurs :

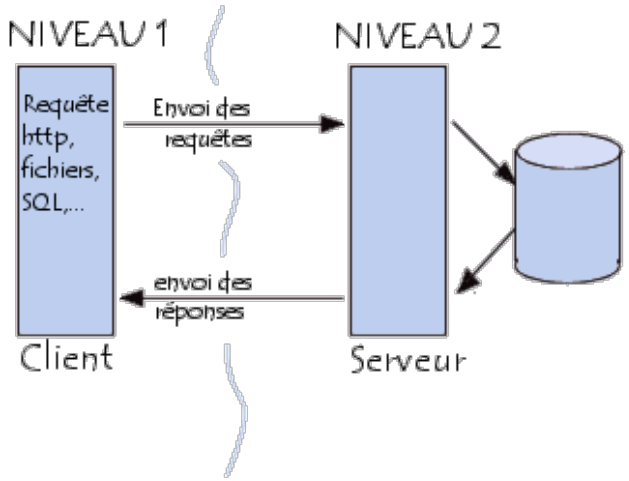


• Le serveur reçoit des **requêtes** de clients pour :

– créer/modifier la base de donnée (seulement les administrateurs)
– Interroger la base de donnée (tous les utilisateurs autorisés).

Architecture deux-tiers

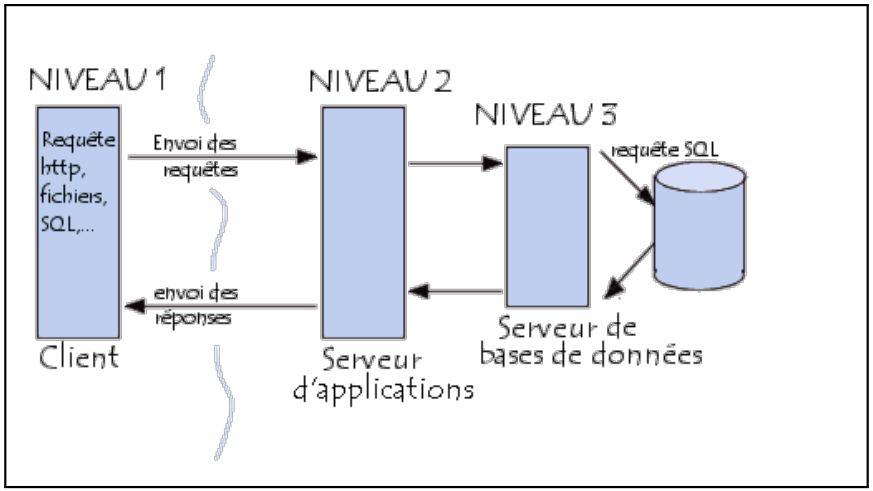
- Une base de donnée peut être conçue selon une **architecture deux tiers**.



- Le client envoie directement ses requêtes au serveurs qui y répond directement. ('tiers' est un anglicisme signifiant 'niveau' ou 'étage').
- Mais pour plus de flexibilité, de sécurité et de performance, on utilise souvent un troisième niveau intermédiaire entre le client et le serveur, contenant l'application qui formule au serveur les requêtes pour en retourner le résultat au client.

Architecture trois tiers

C'est l'**architecture trois tiers**.



- Constituée des 3 niveaux (tiers) :
  1. Le **niveau utilisateur** : c'est la partie interface. Installée sur les ordinateurs 'clients'. Elle permet de formuler les requêtes. L'utilisateur n'a qu'un accès limité à la base de donnée.
  2. Le **niveau applicatif** : c'est la partie logicielle qui fait le lien entre les requêtes de l'utilisateur client et le serveur de base de donnée. En général sur un serveur d'application via un réseau.
  3. Le **niveau base de données** : c'est la partie logicielle qui résout les requêtes, gère de façon optimale les données et procède aux modifications. En général sur un autre serveur du réseau.
- Les 3 installations logicielles sont indépendantes.

Système de gestion de bases de données

- L'accès à une base de donnée et sa gestion s'effectuent à l'aide d'un **système de gestion de bases de données** (en abrégé SGBD). Il résout les requêtes et gère la BdD. La plupart des SGBD utilisent le même langage SQL : Structured Query Language (langage de requête structuré).
- Notamment :
  1. Oracle. Logiciel propriétaire (Oracle corporation). Utilisé dans la gestion des grosses BdD d'entreprise.
  2. MySQL. Freeware, c'est l'un des systèmes de gestion de bases de données les plus répandus, notamment pour les sites web. On peut le coupler à PhpMyAdmin pour bénéficier d'une interface graphique. Disponible à l'adresse : `https://www.mysql.fr`. Installation(s) un peu laborieuse(s).
  3. SQLite. Freeware. Pour la gestion de BdD de tailles limitées. Facile à installer, portable (interface graphique disponible). Parfait pour une utilisation domestique, notamment employé sur les applis smartphones.

     C'est ce dernier SGBD que nous utiliserons.

Vocabulaire des BDD

Une **Base de donnée**	est constituée de tables, contenant les données organisées selon un schéma relationnel.
Une **Table** ou **Relation**	est un ensemble de données organisées vérifiant un même schéma relationnel.
Un **schéma relationnel**	est une suite finie d'attributs (colonnes d'une table).
Un **Attribut**	définit la colonne d'une table : nom (identifiant) et type.
Un **Enregistrement**	est un élément d'une table (une ligne).
Une **Clé**	est un ou plusieurs attributs donnant accès à un seul élément dans une table.
La **Clé primaire**	est la clé choisie parmi toutes les clés possibles.
Une **Clé secondaire**	est toute clé autre que la clé primaire

Théorie : algèbre relationnelle

- Une BDD est constituée de tables (ou relations) chacune définie par un schéma relationnel.
- Un **schéma relationnel** est une suite finie d'attributs : $S = (A_1, A_2, \ldots, A_p)$.
- Chaque **attribut** A_i est doté d'un *type* ; en SQL les types peuvent être : `NULL`, `BOOLEAN`, `INT`, `INT(N)`, `DECIMAL`, `DECIMAL(N)`, `DECIMAL(N,M)`, `FLOAT`, `FLOAT(N)`, `VARCHAR`, `VARCHAR(N)`, `DATE`, `YEAR`, `TIMES`, etc ... (SQLite comprend ces derniers mais ne stocke que sous les types suivants : `NULL`, `INTEGER`, `REAL`, `TEXT`, `BLOB`.)
- Chaque type définit le *Domaine* $DOM(A_i)$ de l'attribut A_i, c'est à dire l'ensemble des valeurs qu'il peut prendre.
- Une relation $R(S)$ de schéma relationnel $S = (A_1, A_2, \ldots, A_p)$, est un sous-ensemble fini du produit cartésien :

$$DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_p)$$

Une relation (ou table) est analogue à un ensemble mathématique à la différence près qu'il est possible d'avoir répétitions. C'est possible grâce l'usage d'un identifiant clé : $R(S) \quad \subset \quad \mathbb{N} \times DOM(A_1) \times \cdots \times DOM(A_p)$.

Théorie : algèbre relationnelle

• On dit que deux relations *ont le même schéma* si ils ont même attributs (noms et types).

• $e \in DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_p)$ est un *enregistrement* d'une relation $R(A_1, A_2, \ldots, A_p)$.

• Si A_k est un attribut, alors $e[A_k]$ désigne la valeur de l'attribut A_k dans l'enregistrement e.

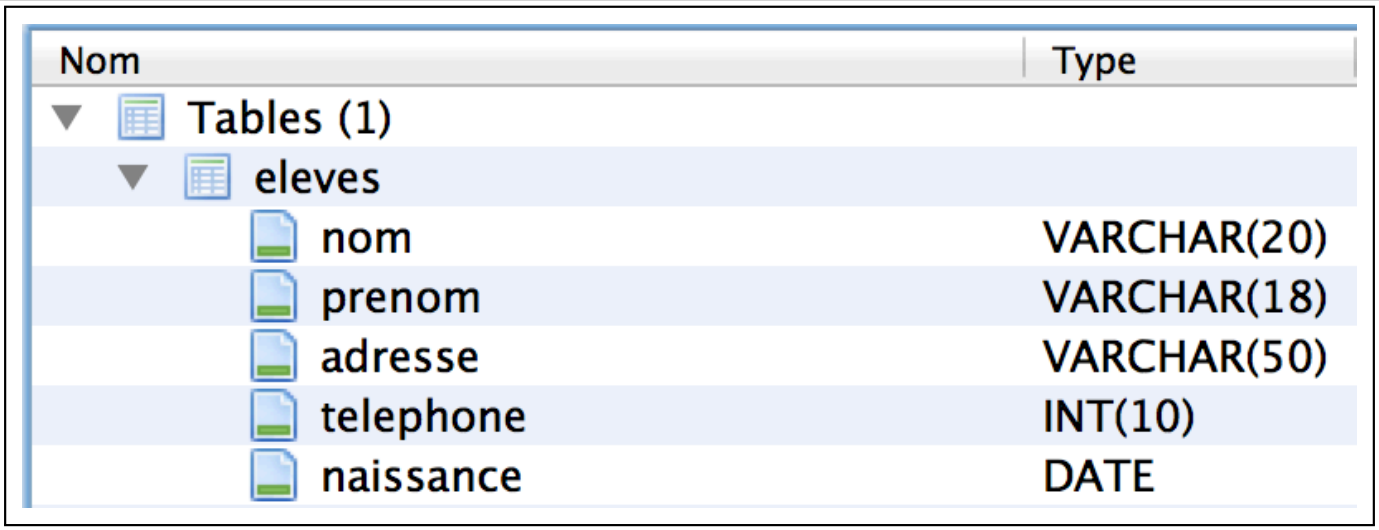
• Si $K = (A'_1, \ldots A'_q)$ est une sous-suite d'attributs de $(A_1, \ldots A_p)$, on note $e[K]$ le q-uplet $\left(e[A'_1], \ldots, e[A'_q]\right)$ des valeurs dans e des attributs $A'_1, \ldots, A'_q$.

• Pour une relation $T \subset DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_p)$, une *clé* est un ensemble C d'attributs, tels que pour tout couple e, e' dans T :

$$e[C] = e'[C] \implies e = e'$$

• Dans une base de donnée nommée `classe` : créons une table `eleves` ayant pour attributs : nom, prénom, adresse, telephone, date de naissance :

```
CREATE TABLE eleves (
    nom VARCHAR(20),
    prenom VARCHAR(18),
    adresse VARCHAR(50),
    telephone INT(10),
    naissance DATE )
```



Par défaut la valeur de chaque attribut est `NULL`.

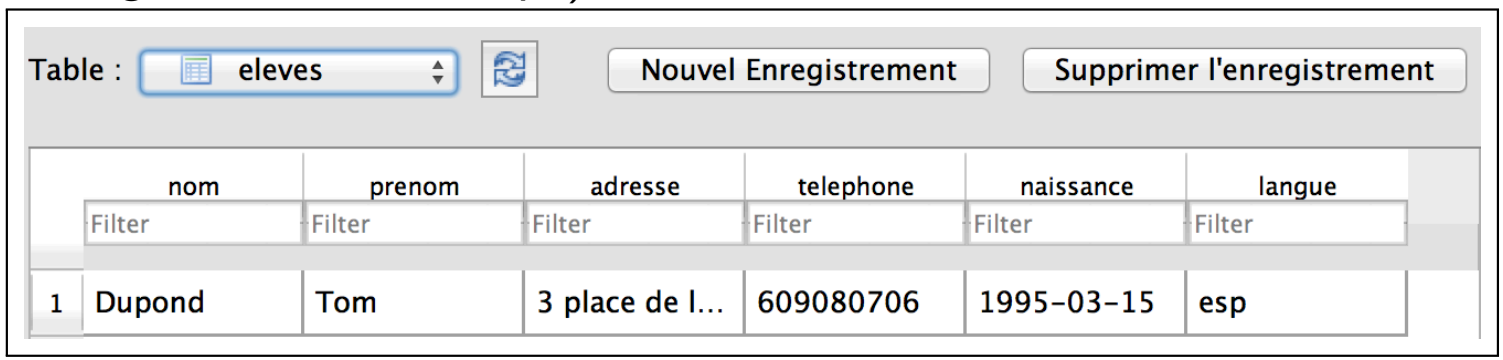
Ajout d'enregistrement

- Ajoutons des enregistrement à la table :

1. A l'aide de la requête `INSERT INTO ... VALUES` :

```
INSERT INTO eleves VALUES ('Dupond', 'Tom', '3 place de la gare',
06609080706, '1995-03-15', 'esp')
```

(dans l'onglet "exécuter le sql")



2. En cliquant sur le bouton 'Nouvel Enregistrement' et en saisissant les données.
(dans l'onglet "parcourir les données").

3. À partir des données d'une autre table, à l'aide de :
`INSERT INTO ... SELECT ... FROM ...`.

• Nous avons saisi manuellement les enregistrements suivants dans la table `eleves` :

	nom	prenom	adresse	telephone	naissance	langue
	Filter	Filter	Filter	Filter	Filter	Filter
1	Dupond	Tom	3 place de la gare	609080706	1995-03-15	esp
2	Charron	Ella	10 rue de la gare	0607080908	1994-12-10	ita
3	Cuvelier	Paul	11 rue Paul Bert	0665544332	1995-09-12	all
4	Dumas	Anne	5 rue de l'oiseau	0667778778	1994-10-01	rus
5	Henri	Thomas	3 quai du port	0607677878	1995-01-24	all
6	Laurin	Theo	4 place de l'Ormeau	0605040321	1995-06-07	esp

• On peut alors afficher la table grâce à la requête :

```
SELECT * FROM eleves
```

qui produit pour résultat :

	nom	prenom	adresse	telephone	naissance	langue
1	Dupond	Tom	3 place de la gare	609080706	1995-03-15	esp
2	Charron	Ella	10 rue de la gare	607080908	1994-12-10	ita
3	Cuvelier	Paul	11 rue Paul Bert	665544332	1995-09-12	all
4	Dumas	Anne	5 rue de l'oiseau	667778778	1994-10-01	rus
5	Henri	Thomas	3 quai du port	607677878	1995-01-24	all
6	Laurin	Theo	4 place de l'Ormeau	605040321	1995-06-07	esp

- On peut ne sélectionner qu'un enregistrement :

```
SELECT * FROM eleves WHERE nom = 'Charron'
```

	nom	prenom	adresse	telephone	naissance	langue
1	Charron	Ella	10 rue de la gare	607080908	1994-12-10	ita

- ou ne sélectionner que certains attributs :

```
SELECT nom, prénom FROM eleves
```

	nom	prenom
1	Dupond	Tom
2	Charron	Ella
3	Cuvelier	Paul
4	Dumas	Anne
5	Henri	Thomas
6	Laurin	Theo

- `... ORDER BY...` permet de les ordonner selon une valeur d'attribut :

```
SELECT nom, prénom, naissance FROM eleves ORDER BY nom
```

	nom	prenom	naissance
1	Charron	Ella	1994-12-10
2	Cuvelier	Paul	1995-09-12
3	Dumas	Anne	1994-10-01
4	Dupond	Tom	1995-03-15
5	Henri	Thomas	1995-01-24

Commandes SQL de manipulation de tables

`CREATE TABLE ma_table (...)`	Création d'une table dans une BDD
`INSERT INTO ma_table VALUES (...)`	Ajout d'un enregistrement
`SELECT * FROM ma_table`	Afficher tous les enregistrements de la table `ma_table`
`SELECT A1, ..., Ap FROM ma_table`	Projection : afficher certaines colonnes de la table `ma_table`
`SELECT ... FROM ... WHERE condition`	Sélection : sélectionne les enregistre--ments vérifiant une condition
`SELECT ... ORDER BY Ak [DESC]`	Ordonner l'affichage selon un attribut
`SELECT ... LIMIT n`	Limitation aux `n` premiers éléments
`SELECT ... LIMIT m, n`	Limitation aux éléments de `m` à `n`

Opérateurs unaires : projection et sélection

- Soit une relation (table) $r(A_1, A_2, \ldots, A_p)$.

Projection suivant un sous-ensemble d'attribut :

- Soit $X = (A'_1, A'_2, \ldots, A'_q)$ une suite finie d'attributs dans $\{A_1, A_2, \ldots, A_p\}$. La **projection** de la relation r suivant X est :

$$\pi_X(r) = \{e[X]; e \in r\}$$

On demande aussi à la projection d'effacer les doublons.

Sélection par une propriété :

- Si on se donne une propriété (ou prédicat) P, la **sélection** (ou **restriction**) de la relation r par la propriété P est :

$$\sigma_P(r) = \{e \in r\,;\ e \text{ satisfait } P\}$$

- **Projection et sélection retournent une relation !**

Ce qui autorise les requêtes composées :

```
SELECT ... FROM (SELECT ... FROM ...) ...
```

- **Renommage** : changer le nom d'un attribut avec `AS`.

Projection :

- Pour projeter : `SELECT DISTINCT attributs FROM`

```
SELECT DISTINCT nom, prénom FROM eleves
```

projette la table `eleves` suivant les attributs `nom, prénom`.

	nom	prenom
1	Dupond	Tom
2	Charron	Ella
3	Cuvelier	Paul
4	Dumas	Anne
5	Henri	Thomas
6	Laurin	Theo

- pour sauvegarder les résultats dans une table en ordonnant par nom :

```
CREATE TABLE t (nom VARCHAR(20), prénom VARCHAR(18)) ;
INSERT INTO t SELECT nom, prénom FROM eleves ORDER BY nom
```

Sélection :

- Pour sélectionner : `SELECT ... FROM ... WHERE ...`

```
SELECT * FROM eleves WHERE naissance > '1995-01-01'
```

Sélection de toutes les lignes des élèves nés au plus tôt en 1995.

	nom	prenom	adresse	telephone	naissance	langue
1	Dupond	Tom	3 place de la gare	609080706	1995-03-15	esp
2	Cuvelier	Paul	11 rue Paul Bert	665544332	1995-09-12	all
3	Henri	Thomas	3 quai du port	607677878	1995-01-24	all
4	Laurin	Theo	4 place de l'Ormeau	605040321	1995-06-07	esp

```
SELECT nom, prenom FROM eleves WHERE naissance > '1995-01-01'
     AND nom LIKE 'C%'
```

Sélection des noms et prénoms des élèves nés après 1995 dont le nom commence par un C.

	nom	prenom
1	Cuvelier	Paul

Sélection :

- Pour sélectionner : `SELECT attributs FROM table WHERE conditions`

Les conditions peuvent être constituées :

1. Opérateurs de comparaison : =, <, >, <=, etc...
2. Connecteurs logiques : `AND`, `OR`, `NOT`, `XOR` et parenthèses.
3. `attribut BETWEEN a AND b` $\Longleftrightarrow$ `attribut >= a AND attribut <= b`
4. Expressions régulières : `LIKE chaine_de_caractère`, avec :
   - 4.1 `'_'` remplace n'importe quel caractère
   - 4.2 `'%'` remplace n'importe quelle chaîne de caractère
5. Prédicat ensemblistes : `EXISTS()`, `IN()`
   - 5.1 `EXISTS(table)` est vérifié lorsque `table` est non vide. Le plus souvent `table` est `SELECT .... FROM ...` : le résultat d'une sélection.
   - 5.2 `attribut IN (liste)` est vérifié lorsque `attribut` apparait dans `liste`. Le plus souvent `liste` est une liste de valeurs `(1,2,3,...)` ou une liste résultat d'une sélection : `SELECT attribut FROM ....`

- On peut aussi ajouter :

`ORDER BY attribut [DESC]` ou `LIMIT [m,] n`

en fin de commande pour ordonner/limiter les résultats.

Fonctions d'agrégations

• Une fonction d'agrégation est une application qui a une table associe un nombre. Le plus souvent pour compter, faire la moyenne d'un attribut, etc..., selon une colonne.

• Les principales fonctions d'agrégation :

`COUNT`	pour compter le nombre d'enregistrements distincts d'une table

`AVG`	valeur moyenne d'une colonne de type numérique d'une table
`MIN`	valeur minimale d'une colonne de type numérique d'une table
`MAX`	valeur maximale d'une colonne de type numérique d'une table
`SUM`	somme d'une colonne de type numérique d'une table

• Exemple : Compter les élèves (en paramètre les attributs à compter, les `NULL` ne sont pas comptabilisés)

```
SELECT COUNT(*) FROM eleves
```

	COUNT(*)
1	6

La table contient 6 enregistrements (= élèves).

Fonctions d'agrégations

• Une fonction d'agrégation retourne une seule ligne. Avec la commande `GROUP BY` on retourne plusieurs lignes en regroupant les mêmes valeurs d'un attribut et en effectuant l'agrégation sur chaque groupe :

```
SELECT COUNT(*) FROM eleves GROUP BY langue
```

On compte le nombre d'élèves par groupe de langue.

	COUNT(*)	langue
1	2	all
2	2	esp
3	1	ita
4	1	rus

• `GROUP BY` peut être suivi de `HAVING condition` pour requérir une condition dans la constitution des groupes :

```
SELECT COUNT(*) FROM eleves GROUP BY langue HAVING langue = 'esp'
```

ne comptera que le groupe des élèves faisant espagnol. Même résultat avec :

```
SELECT COUNT(*) FROM (SELECT * FROM eleves WHERE langue = 'esp')
```

Différence avec `WHERE` : `HAVING` peut être suivie d'une condition invoquant une fonction d'agrégation : `MAX()`, `MIN()`, `SUM()`, `COUNT()`.

- Créons une nouvelle table dans notre BDD contenant les notes des élèves :

```
CREATE TABLE notes(nom VARCHAR(20), prenom VARCHAR(18),
math DECIMAL(4,2), physique DECIMAL(4,2), français DECIMAL(4,2),
anglais DECIMAL(4,2),
PRIMARY KEY (nom, prénom))
```

nom	VARCHAR(20)	`nom` VARCHAR(20)
prenom	VARCHAR(18)	`prenom` VARCHAR(18)
math	DECIMAL(4,2)	`math` DECIMAL(4,2)
physique	DECIMAL(4,2)	`physique` DECIMAL(4,2)
français	DECIMAL(4,2)	`français` DECIMAL(4,2)
anglais	DECIMAL(4,2)	`anglais` DECIMAL(4,2)

Que l'on remplit ensuite.

	nom	prenom	math	physique	français	anglais
	Filter	Filter	Filter	Filter	Filter	Filter
1	Dupond	Tom	10	12	14	15
2	Dumas	Anne	10	12	14	15
3	Laurin	Theo	11	8	13	12
4	Charron	Ella	9	10	11	7
5	Henri	Thomas	8	11	7	6
6	Cuvelier	Paul	5	10	8	12

- La moyenne en maths s'obtient par la requête :

```
SELECT AVG(math) FROM notes
```

- La moyenne générale de la classe s'obtient par la requête :

```
SELECT (AVG(math)+AVG(physique)+AVG(français)+AVG(anglais)) / 4.0
    FROM notes
```

On affiche la moyenne générale d'un élève par :

```
SELECT nom, (math+physique+français+anglais)/4.0 FROM notes
    WHERE nom = 'Dumas'
```

La moyenne générale des élèves s'obtient par la requête :

```
SELECT nom, (math+physique+français+anglais)/4.0 FROM notes
```

	nom	(math+physique+français+anglais)/4
1	Dupond	12
2	Dumas	12
3	Laurin	11
4	Charron	9
5	Henri	8
6	Cuvelier	8

Ou encore, pour un résultat plus esthétique effectuer un renommage avec AS :

```
SELECT nom, (math+physique+français+anglais)/4.0 AS moyenne FROM notes
```

	nom	moyenne
1	Dupond	12
2	Dumas	12
3	Laurin	11
4	Charron	9
5	Henri	8
6	Cuvelier	8

- Le nom du major de promo s'obtient par la requête :

```
SELECT nom, (math+physique+français+anglais)/4.0 AS moyenne
FROM notes ORDER BY moyenne DESC LIMIT 1
```

- Les noms des élèves ayant une note en maths supérieure à la moyenne de la classe :

```
SELECT nom FROM notes WHERE math >= (SELECT AVG(math) FROM notes)
```

Une requête retournant une seule valeur peut être traité comme une valeur !

• A l'aide des tables `eleves` et `notes` et d'une requête composée retourner le nom et numéro de téléphone des élèves n'ayant pas la moyenne générale :

```
SELECT nom, telephone FROM eleves
    WHERE nom IN (SELECT nom FROM notes
        WHERE (math+physique+français+anglais)/4.0 < 10)
```

• Retourner le nom et numéro de téléphone des élèves ayant au moins une note inférieure à la moyenne :

```
SELECT nom, telephone FROM eleves
    WHERE NOT(nom IN (SELECT nom FROM notes
        WHERE math>= 10 AND physique >= 10 AND français >= 10
            AND anglais >=10))
```