

# Les bases de données

## 1 Les bases de données relationnelles

Lorsque l'on doit gérer un grand nombre d'informations avec un système informatique, on a souvent recours à une **base de données**. Dans une base de données relationnelle, l'information est stockée dans des **tables**

Une table est un tableau dont les colonnes sont appelées **attributs** et chaque ligne est une **relation** (ou **enregistrement**).

Par exemple, si l'on veut stocker les informations concernant les étudiants de TS11 dans une base de données, on peut créer une table appelée `etudiants` contenant les attributs (colonnes) suivants :

- nom,
- prenom,
- anniversaire,
- mail

Un enregistrement (ou relation) de cette table est constitué d'un groupe de valeurs correspondant à chacun des attributs, par exemple : (Hutasse, Émilien, 10/10/2000, mimi2k@hotmail.fr).

Pour une table donnée une **clé** correspond à un groupe d'attributs pour lesquels les valeurs sont différentes pour chaque relation. Par exemple le groupe d'attributs (nom,prenom) constituent une clé pour la table `etudiants` car il n'y a pas deux étudiants qui ont le même nom et le même prénom. L'attribut `nom` constitue à lui seul une clé car tous les étudiants ont un nom différent. Une clé constituée d'un seul attribut s'appelle une **clé primaire**.

L'attribut `prenom` n'est pas une clé pour la table `etudiants` car il y a des prénoms communs à plusieurs étudiants.

L'ensemble des valeurs autorisées pour un attribut est appelé **domaine**. Le domaine de l'attribut `nom` est une chaîne de caractères, celui de `anniversaire` est une date.

Le **schéma** d'une relation est constitué de l'ensemble des attributs et de leur domaine. Par exemple le schéma de la table `etudiants` est :

((nom,CHAINE) (prenom,CHAINE) (anniversaire,CHAINE) (mail,CHAINE)).

## 2 Le langage SQL

Le langage **SQL** (Structured query language) est un langage informatique servant à exploiter une base de données relationnelle. Ce langage permet d'extraire, de modifier ou d'ajouter des informations dans une base de données relationnelle.

### 2.1 Extraire l'information

Pour accéder aux valeurs de certains attributs des enregistrements d'une table on peut utiliser la commande :

**SELECT** attrs **FROM** tbl. Par exemple la commande

```
SELECT nom,prenom FROM etudiants;
```

renvoie les noms et prénoms de tous les étudiants.

On peut également filtrer les informations récupérées en ajoutant l'instruction **WHERE** `cond` où `cond` est une condition portant sur les attributs sélectionnés. Par exemple :

```
SELECT nom FROM etudiants WHERE prenom='Lucas';
```

renvoie la liste des noms des étudiants dont le prénom est Lucas.

Plusieurs conditions peuvent être combinées avec les opérations **AND** (et) ou **OR** (ou). Par exemple on peut affiner la sélection précédente en utilisant :

```
SELECT nom FROM etudiants WHERE prenom='Lucas' OR prenom='Simon';
```

Les résultats d'une requête peuvent être triés en fonction des valeurs d'une colonne grâce à l'instruction **ORDER BY**. Par exemple :

```
SELECT nom, prenom FROM etudiants ORDER BY anniversaire;
```

Pour limiter le nombre de résultats renvoyés par une requête, on utilise le mot-clé **LIMIT**. Par exemple si on veut déterminer l'étudiant le plus jeune on utilisera la requête :

```
SELECT nom, prenom FROM etudiants ORDER BY anniversaire DESC LIMIT 1;
```

## 2.2 Combiner des informations : JOIN

L'exemple précédent est extrêmement simpliste, il ne contient qu'une seule table. Lorsque l'information est plus complexe, on a souvent recours à plusieurs tables pour l'organiser.

On peut considérer l'exemple d'un logiciel de gestion de notes d'une classe qui utiliserait une base de données contenant au moins trois tables :

Id	Nom	Date	Coefficient
1	DS1	16/09/2017	4
2	TP1	13/09/2017	0.5
3	TP2	20/09/2017	0.5
⋮	⋮	⋮	

  

Id	Nom	Prenom
1	AGNUS	David
2	AIT FASKA	Saïd
3	BEGUINOT	Alan
⋮	⋮	⋮

  

IdEtud	IdDevoir	Note
1	1	7.5
2	1	13
3	1	7.5
⋮	⋮	⋮
1	2	15
2	2	14
3	2	15
⋮	⋮	⋮

- une table *Devoirs* qui contient la liste des devoirs et leurs attributs;
- une table *Etudiants* qui contient la liste des noms des étudiants;
- une table *Notes* qui contient les notes de chaque étudiant pour chaque devoir.

On peut alors combiner les informations de plusieurs tables pour en extraire les notes obtenues au DS1 :

```
SELECT IdEtud, Note FROM Notes JOIN Devoirs ON IdDevoir=Id WHERE Nom='DS1';
```

Ou pour avoir également les noms des étudiants :

```
SELECT Nom, Prenom, Note
FROM Notes
JOIN Devoirs ON IdDevoir=Devoirs.Id
JOIN Etudiants ON IdEtud=Etudiants.Id
WHERE Devoirs.Nom='DS1';
```

à noter que comme il y a deux colonnes qui portent le même nom, on les préfixe avec le nom de la table : la colonne Id de la table *Devoirs* s'appelle *Devoirs.Id*

## 2.3 Agréger des informations

Le langage SQL permet également d'agréger des informations, par exemple de calculer la moyenne d'une liste de nombre (**AVG**), leur somme (**SUM**), de déterminer le plus grand ou le plus petit élément d'une liste (**MAX** et **MIN**) ou de compter le nombre d'éléments d'une liste (**COUNT**)

En reprenant l'exemple précédent, la requête suivante permet de déterminer la moyenne du TP1 :

```
SELECT AVG(Note) FROM Notes WHERE IdDevoir=2;
```

**Attention :** On ne peut pas dans une même requête mélanger des colonnes agrégées et des colonnes non agrégées. Par exemple on ne pourra pas écrire :

```
SELECT IdEtud, MAX(Note) FROM Notes WHERE IdDevoir=1;
```

Pour trouver l'étudiant qui a eu la meilleur note au devoir 1. Pour trouver cette information on pourra plutôt utiliser :

```
SELECT IdEtud, Note FROM Notes WHERE IdDevoir=1 ORDER BY Note DESC LIMIT 1;
```

Pour appliquer une fonction d'agrégation à plusieurs groupes distincts, on utilise la commande **GROUP BY**. Par exemple pour déterminer les moyennes de chaque devoirs on peut effectuer la requête :

```
SELECT IdDevoir,AVG(Note) AS moyenne FROM Notes GROUP BY IdDevoir;
```

Pour que la requête ne renvoie que les moyennes supérieures à 10 on utilise la requête :

```
SELECT IdDevoir,AVG(Note) AS moyenne  
FROM Notes  
GROUP BY IdDevoir HAVING moyenne>10;
```

Il faut noter que dans ce cas on n'utilise pas la commande **WHERE**

## 2.4 Renommer une colonne ou une table : AS

Dans une commande on peut vouloir par soucis de concision, renommer une colonne ou une table avec un nom plus court, ou plus pratique. Par exemple, on peut ré-écrire la requête précédente comme :

```
SELECT Nom, Prenom, Note  
FROM Notes  
JOIN Devoirs AS d ON IdDevoir=d.Id  
JOIN Etudiants AS e ON IdEtud=e.Id  
WHERE Devoirs.Nom='DS1';
```

## 2.5 Ne conserver que les enregistrements différents : DISTINCT

Pour qu'une commande select ne renvoie que les enregistrements différents on peut utiliser la commande **SELECT DISTINCT**. Par exemple dans la table Etudiants on peut afficher les différents prénoms des étudiants avec la requête :

```
SELECT DISTINCT prenom FROM Etudiants;
```

## 2.6 Filtrer une chaîne de caractères : LIKE

On peut filtrer les résultats d'une requête selon une colonne contenant une chaîne de caractères avec la commande **LIKE**. Par exemple la requête

```
SELECT nom, prenom FROM Etudiants WHERE nom LIKE "B%"
```

retourne le nom et le prénom des étudiants dont le nom commence par un "B". Le caractère % remplace n'importe quelle sous-chaîne de caractères. Pour afficher le nom et le prénom des étudiants dont le nom contient un "O", on utilisera la requête

```
SELECT nom, prenom FROM Etudiants WHERE nom LIKE "%O%"
```