

JDBC

(1ere partie)

Bibliographies: Pour la première partie de ce cours, une revue littéraire a été réalisée à partir des documents mentionnés ci-dessous. Comme la première partie n'est qu'un bref aperçu sur les Bases de données et SQL, je n'ai extrait donc que quelques transparents par ci par là. De ce fait, je vous encourage à jeter un coup d'œil sur chacun des documents pour avoir une vue plus profonde sur la question. Le document cdb.pdf devra passer en premier car c'est des notes de cours avec des explications détaillées.

Site web officiel de SUN sur les JDBC: <http://java.sun.com/products/jdbc>

Les Bdds & SQL

<http://cortes.cnam.fr:8080/BDA/DOC/cbd.pdf>

<http://deptinfo.unice.fr/~grin/messupports/trsgbd.pdf>

Plusieurs transparents ont été extraits de ce lien:

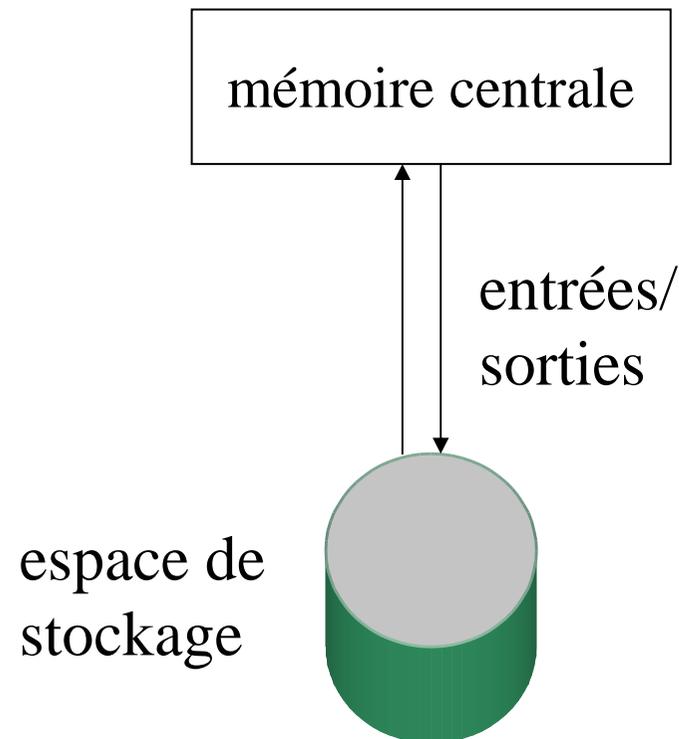
<http://cortes.cnam.fr:8080/BDA/SLIDES/>

Définitions

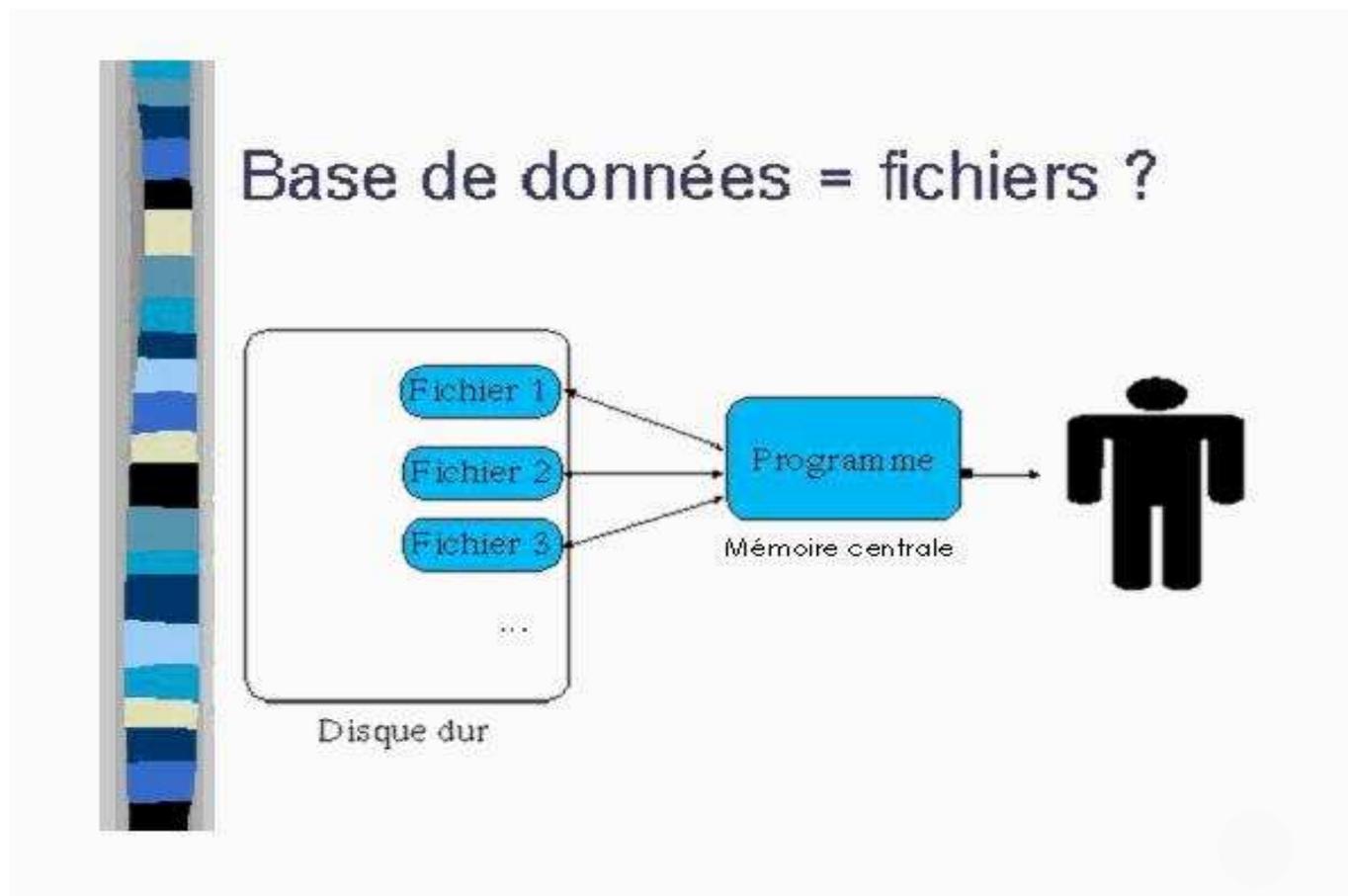
"Une base de données est un ensemble structuré de données enregistrées dans un ordinateur et accessibles de façon sélective par plusieurs utilisateurs."

"Une base de données =

- un (gros) ensemble d'informations
- stockées sur disque:
 - o) persistance
 - o) espace de stockage"



Cette définition suppose que nous pouvons organiser cette base en un (ou plusieurs) fichier(s) stockés sur mémoire.



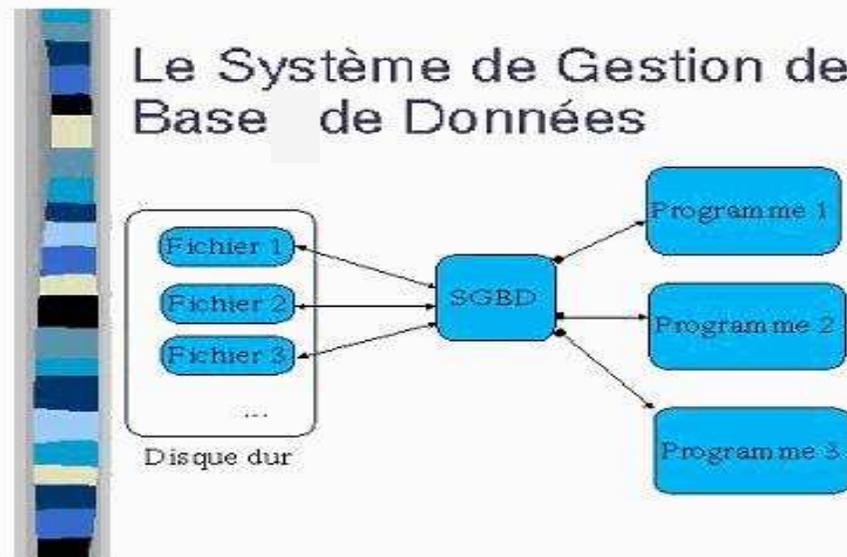
Cette approche pose certains problèmes:

1. "Lourdeur d'accès aux données. En pratique, pour chaque accès, même le plus simple, il faudrait écrire un programme."
2. "Manque de sécurité. Si tout programmeur peut accéder directement aux fichiers, il est impossible de garantir la sécurité et l'intégrité des données."
3. "Pas de contrôle de concurrence. Dans un environnement où plusieurs utilisateurs accèdent aux mêmes fichiers, des problèmes de concurrence d'accès se posent."

Solution => SGBD!

Systeme de Gestion de Base de Données (SGBD) (Database Management System: DBMS)

"Un logiciel qui permet d'interagir avec une base de données s'appelle un système de gestion de base de données"

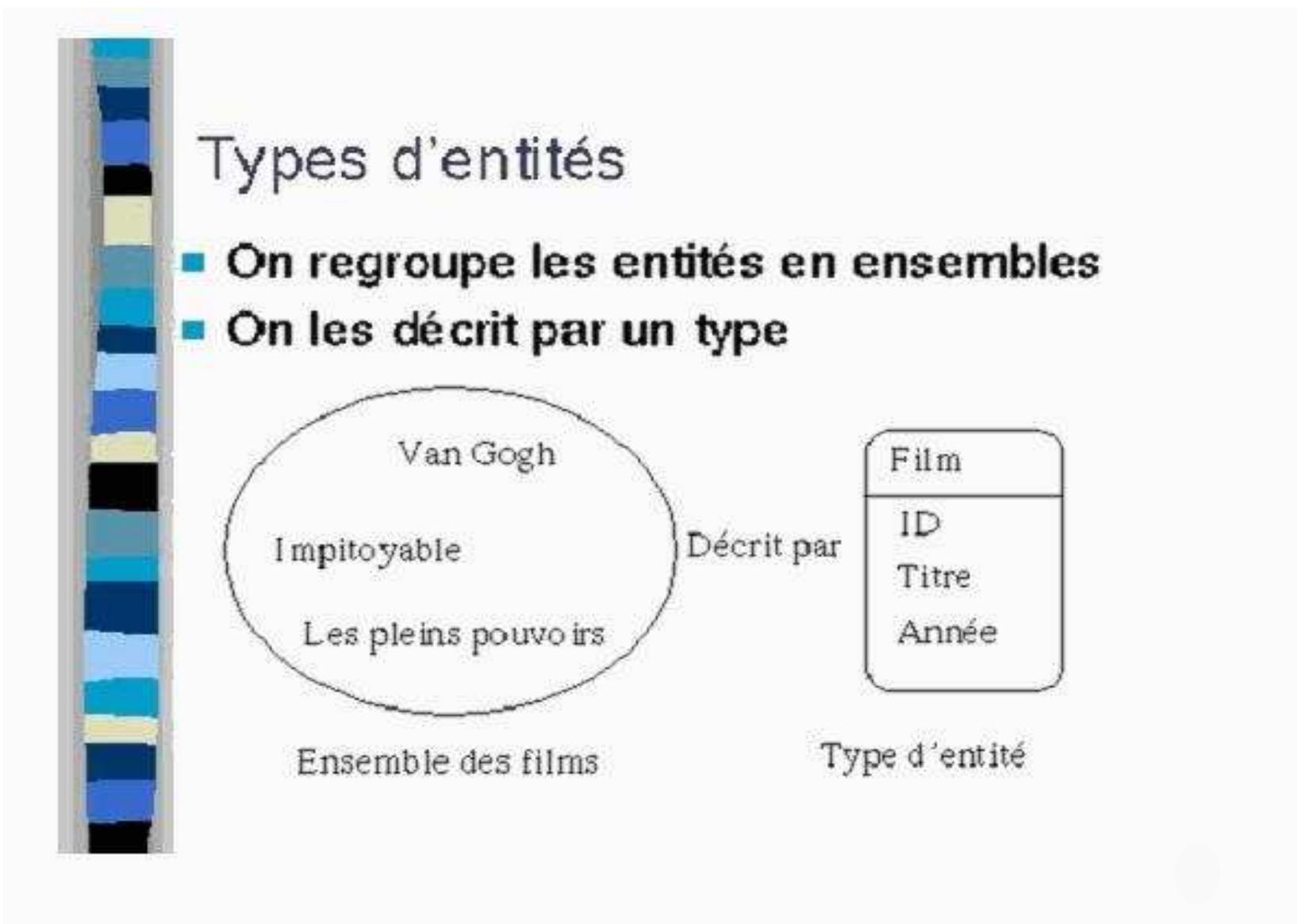


Les fonctionnalités d'un SGBD sont

- "Décrire les données qui seront stockées"
- "Manipuler ces données (ajouter, modifier, supprimer des informations)"
- "Consulter les données et traiter les informations obtenues (sélectionner, trier, calculer, agréger,...)"
- "Définir des contraintes d'intégrité sur les données (contraintes de domaines, d'existence,...)"
- "Définir des protections d'accès (mots de passe, autorisations,...)"

- "Résoudre les problèmes d'accès multiples aux données (blocages, interblocages)"
- "Prévoir des procédures de reprise en cas d'incident (sauvegardes, journaux,...)"

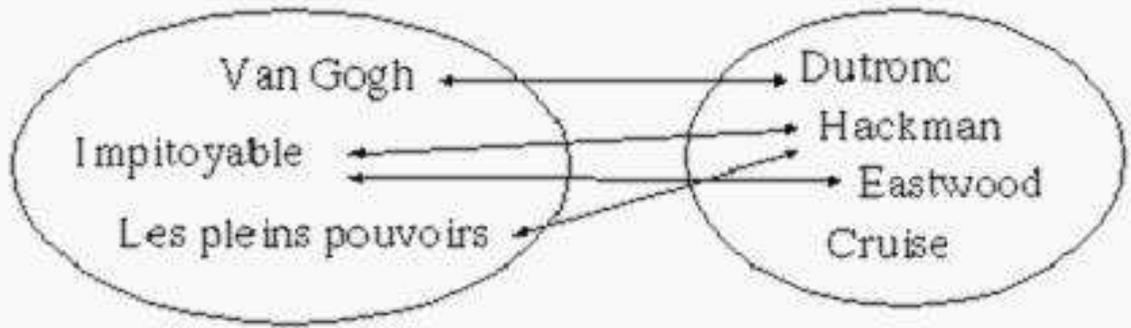
"En résumé, un SGBD est destiné à gérer un gros volume d'informations, persistantes (années) et fiables (protection sur pannes), partageables entre plusieurs utilisateurs et/ou programmes et manipulées indépendamment de leur représentation physique."





Associations entre entités

- Une relation entre ensembles d'entités
- Exemple: « Acteur **JOUE DANS** Film »



Ensemble des films

Ensemble des artistes



Utilisation d'un SGBD: opérations sur les données

- **Quatre opérations d'accès aux données:**
 - Insertion
 - Destruction
 - Modification
 - Recherche
- **Le langage d'accès est SQL. On l'utilise pour dialoguer avec le SGBD.**





Administration d'un SGBD

- **A partir des commandes SQL, le SGBD détermine automatiquement**
 - Où et comment stocker les données
 - Comment effectuer les mises à jour ou les recherches
- **Rôle de l'administrateur:**
 - Restreindre les droits d'accès
 - Garantir la disponibilité du système
 - Améliorer les performances





En résumé : que doit-on savoir?

■ Conception

- Décrire une BD en fonction d'un besoin
- Interpréter correctement une description

■ Utilisation

- Créer correctement une base relationnelle
- Savoir interroger et mettre à jour cette BD

■ Administration

- Comprendre le fonctionnement interne du SGBD

Le modèle relationnel

Une table relationnelle



Nom d'attribut

Films

Nom de la table

titre	année	nomMES	prénomMES	néEn
Alien	1979	Scott	Ridley	1953
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Titanic	1997	Cameron	James	1954
Sacrifice	1936	Tarkovski	Andrei	1932

*Ligne
ou
tuple*

Colonne

Base de données relationnelle

- **Une BD relationnelle = un ensemble de tables**
 - relatives à une même application
 - Ex: la base « Officiel des spectacles »
 - chaque table est **décrite** par son **schéma**
 - l'ensemble des schémas des tables constitue **le schéma de la BD**
- **Il y a des bons et des mauvais schémas !**

Bon et mauvais schémas ?

Films

titre	année	nomMES	prénomMES	néEn
Alien	1979	Scott	Ridley	1953
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Titanic	1997	Cameron	James	1954
Sacrifice	1936	Tarkovski	Andrei	1932

- **Redondance <=> pas bon !**

Risque d'incohérence:

Films

titre	année	nomMES	prénomMES	néEn
Alien	1979	Scott	Ridley	1953
Vertigo	1958	Hitchcock	Gaston	1900
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Titanic	1997	Cameron	James	1954
Sacrifice	1936	Tarkovski	Andrei	1932

- -> **des anomalies apparaissent en insertion ou mises à jour**



Anomalies de destruction

- **Je supprime des films ?**

titre	année	nomMES	prénomMES	néEn
Alien	1979	Scott	Ridley	1953
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Titanic	1997	Cameron	James	1954
Sacrifice	1936	Tarkovski	Andrei	1932

- **-> j'ai aussi supprimé des metteurs en scène !**

Première solution : faire deux tables

titre	année
Alien	1979
Vertigo	1958
Psychose	1960
Kagemusha	1980
Volte-face	1997
Titanic	1997
Sacrifice	1936

nomMES	prénomMES	néEn
Scott	Ridley	1953
Hitchcock	Alfred	1899
Kurosawa	Akira	1910
Woo	John	1946
Cameron	James	1954
Tarkovski	Andrei	1932

■ Questions:

- Qu'est-ce qui permet de dire qu'on n'a plus de redondance ??
- Plus de lien Film <-> Metteur en scène

Meilleure solution : avec identifiants

Clé étrangère *Clé primaire*

titre	année	IdMES	ID	nomMES	prénomMES
Alien	1979	100	100	Scott	Ridley
Vertigo	1958	101	101	Hitchcock	Alfred
Psychose	1960	101	102	Kurosawa	Akira
Kagemusha	1980	102	103	Woo	John
Volte-face	1997	103	104	Cameron	James
Titanic	1997	104	105	Tarkovski	Andrei
Sacrifice	1936	105			

■ C'est un bon schéma !

- Chaque donnée à sa place, une seule fois
- Possibilité de reconstruire le lien



Une démarche systématique

- **On part du schéma Entité/Association**
 - Les principaux choix ont été faits à ce niveau
- **On applique des règles de passage**
 - On obtient un schéma relationnel
- **Commandes de création des tables**
 - attributs
 - clés primaires
 - clés étrangères

Notation

- Un schéma de table est noté de la manière suivante :

Film (idFilm, titre, année, idMES)

↑ ↑ ↑

Nom *Clé* *Clé*

table *primaire* *étrangère*

- Pour l'instant on ne s'occupe pas du type des attributs

Le langage SQL

"SQL (Structured Query Language ou bien Langage de requête structuré) est un langage déclaratif, permet d'interroger une base de données sans se soucier de:

la représentation interne (physique) des données, de leur localisation, des chemins d'accès, ou des algorithmes nécessaires."

"SQL s'adresse à une large communauté d'utilisateurs potentiels (pas seulement des informaticiens) et constitue un des atouts les plus spectaculaires (et le plus connu) des SGBDR (R pour Relationnelle)."

"SQL peut être utilisé:

- manière interactive,
- en association avec des interfaces graphiques,
- ou, très généralement, des langages de programmation."

"SQL ne permet pas de faire de la programmation au sens courant du terme (faire une boucle par exemple) et doit donc être associé avec un langage comme le C, le COBOL ou JAVA pour réaliser des traitements complexes accédant à une base de données."

Requêtes simples SQL

NomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

La table **Station**

Les stations
et leurs activités

NomStation	Libellé	Prix
Venusa	Voile	150
Venusa	Plongée	120
Farniente	Plongée	130
Passac	Ski	200
Passac	Piscine	20
Santalba	Kayac	50

La table **Activité**

```
SELECT NomStation FROM Station WHERE region = Antilles
```

Ce premier exemple montre la structure de base d'une requête SQL, avec les trois clauses **SELECT**, **FROM** et **WHERE**.

FROM indique la (ou les) tables dans lesquelles on trouve les attributs utiles à la requête. Un attribut peut être utile de deux manières (non exclusives) :

- (1) on souhaite afficher son contenu,
- (2) on souhaite qu'il ait une valeur particulière (une constante ou la valeur d'un autre attribut).

Il est obligatoire avec **SELECT**.

SELECT indique la liste des attributs constituant le résultat.

WHERE indique les conditions que doivent satisfaire les n-uplets de la base pour faire partie du résultat.

Pour afficher l'intégralité d'une table, et avoir ainsi toutes les lignes (on omet la clause **WHERE**), et toutes les colonnes, on peut au choix lister tous les attributs ou utiliser le caractère ***** qui a la même signification.

SELECT * FROM NomTable

*SELECT * FROM Station*

* remplace tous les attributs (champs) de la table Station. de ce fait la table station est reproduite.

SELECT NomStation, capacité FROM Station

On récupère les deux colonnes de la table Station.

SELECT NomStation FROM Station WHERE region = Antilles

WHERE critère

NomStation
Venusa
Santalba

```
SELECT libelle, prix / 6.56, Cours de l'euro = , 6.56  
FROM Activite  
WHERE nomStation = Santalba
```

libelle	prix / 6.56	Cours de l'euro =	6.56
Kayac	7.62	Cours de l'euro =	6.56

```
SELECT libelle FROM Activite
```

donnera autant de lignes dans le résultat que dans la table Activite.

libelle
Voile
Plongee
Plongee
Ski
Piscine
Kayac

Tri du résultat

Nous utilisons la clause **ORDER BY** pour trier les résultats d'une requête. Cette clause doit être suivie de la liste des attributs servant de critère au tri. Exemple :

```
SELECT * FROM Station ORDER BY tarif, nomStation
```

"Trie, en ordre ascendant, les stations par leur tarif, puis, pour un même tarif, présente les stations selon l'ordre lexicographique. Pour trier en ordre descendant, on ajoute le mot-clé **DESC** après la liste des attributs."

WHERE

"Dans la clause **WHERE**, on spécifie une condition booléenne portant sur les attributs des relations du **FROM**. On utilise pour cela de manière standard le **AND**, le **OR**, le **NOT** et les parenthèses pour changer l'ordre de priorité des opérateurs booléens. Par exemple :

```
SELECT nomStation, libelle  
FROM Activite  
WHERE nomStation = Santalba AND (prix > 50 AND prix < 120)
```

Les opérateurs de comparaison: \leq \geq $<$ $==$ $>$ $<>$ $(!=)$

Pour obtenir une recherche par intervalle, on peut également utiliser le mot-clé BETWEEN. La requête précédente est équivalente à :

```
SELECT nomStation, libelle  
FROM Activite  
WHERE nomStation = Santalba AND prix BETWEEN 50 AND 120
```