

### Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.

### Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.
- Une **base de données** est un ensemble de tables.

## Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.
- Une **base de données** est un ensemble de tables.
- Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.

## Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.
- Une **base de données** est un ensemble de tables.
- Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.
- Une **clé primaire** est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

## Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.
- Une **base de données** est un ensemble de tables.
- Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.
- Une **clé primaire** est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

On utilisera souvent des clés primaire ayant un seul attribut

## Définitions

- Une **table** est un tableau à deux dimensions, les colonnes sont appelés **attributs** et les lignes **enregistrements**.
- Une **base de données** est un ensemble de tables.
- Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.
- Une **clé primaire** est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.  
On utilisera souvent des clés primaire ayant un seul attribut
- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.

# C17 Introduction aux bases de données

## 1. Généralités

### Exemple

Extrait de la table Medals de la base de données olympics1976-2008.db :

Id	City	Year	Sport	Discipline	Event	Athlete	Gender	Country	Medal
286	Montreal	1976	Athletics	Athletics	110m hurdles	DRUT, Guy	Men	France	Gold
194	Montreal	1976	Athletics	Athletics	100m	BORZOV, Valery	Men	Soviet Union	Bronze
13810	Beijing	2008	Athletics	Athletics	deathlon	CLAY, Bryan	Men	United States	Gold
3455	Los Angeles	1984	Fencing	Fencing	épée individual	BOISSE, Philippe	Men	France	Gold

## Exemple

Extrait de la table Medals de la base de données olympics1976-2008.db :

Id	City	Year	Sport	Discipline	Event	Athlete	Gender	Country	Medal
286	Montreal	1976	Athletics	Athletics	110m hurdles	DRUT, Guy	Men	France	Gold
194	Montreal	1976	Athletics	Athletics	100m	BORZOV, Valery	Men	Soviet Union	Bronze
13810	Beijing	2008	Athletics	Athletics	decathlon	CLAY, Bryan	Men	United States	Gold
3455	Los Angeles	1984	Fencing	Fencing	épée individual	BOISSE, Philippe	Men	France	Gold

- City est un attribut, son domaine est l'ensemble des noms de villes ayant accueilli les JO sur la période 1976-2008.



### Exemple

Extrait de la table Medals de la base de données olympics1976-2008.db :

Id	City	Year	Sport	Discipline	Event	Athlete	Gender	Country	Medal
286	Montreal	1976	Athletics	Athletics	110m hurdles	DRUT, Guy	Men	France	Gold
194	Montreal	1976	Athletics	Athletics	100m	BORZOV, Valery	Men	Soviet Union	Bronze
13810	Beijing	2008	Athletics	Athletics	decathlon	CLAY, Bryan	Men	United States	Gold
3455	Los Angeles	1984	Fencing	Fencing	épée individual	BOISSE, Philippe	Men	France	Gold

- City est un attribut, son domaine est l'ensemble des noms de villes ayant accueilli les sc jo sur la période 1976-2008.
- Exemple d'enregistrement
-

### Exemple

Extrait de la table Medals de la base de données olympics1976-2008.db :

Id	City	Year	Sport	Discipline	Event	Athlete	Gender	Country	Medal
286	Montreal	1976	Athletics	Athletics	110m hurdles	DRUT, Guy	Men	France	Gold
194	Montreal	1976	Athletics	Athletics	100m	BORZOV, Valery	Men	Soviet Union	Bronze
13810	Beijing	2008	Athletics	Athletics	decathlon	CLAY, Bryan	Men	United States	Gold
3455	Los Angeles	1984	Fencing	Fencing	épée individuel	BOISSE, Philippe	Men	France	Gold

- City est un attribut, son domaine est l'ensemble des noms de villes ayant accueilli les JO sur la période 1976-2008.
- Exemple d'enregistrement
- Id est un numéro unique pour chaque enregistrement, et peut donc servir de clé primaire.

### Schéma d'une table

Le schéma d'une table est la liste de ses attributs avec leur domaine. On souligne le (ou les) attributs formant la clé primaire.

### Exemple

### Schéma d'une table

Le schéma d'une table est la liste de ses attributs avec leur domaine. On souligne le (ou les) attributs formant la clé primaire.

### Exemple

Le schéma relationnel de la table Medals peut s'écrire :

### Schéma d'une table

Le schéma d'une table est la liste de ses attributs avec leur domaine. On souligne le (ou les) attributs formant la clé primaire.

### Exemple

Le schéma relationnel de la table Medals peut s'écrire :

### Schéma d'une table

Le schéma d'une table est la liste de ses attributs avec leur domaine. On souligne le (ou les) attributs formant la clé primaire.

### Exemple

Le schéma relationnel de la table Medals peut s'écrire :

**Medals** (id : INT, City : TEXT, Year : INT, Sport : TEXT,...)

### Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :

### Exemples

### Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :  
`SELECT * FROM table`

### Exemples



## Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :  
`SELECT * FROM table`
- Pour récupérer simplement les champs `champ1`, `champ2`, ... on utilise :

## Exemples

## Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :  
`SELECT * FROM table`
- Pour récupérer simplement les champs `champ1`, `champ2`, ... on utilise :  
`SELECT champ1, champ2, ... FROM table`

## Exemples

## Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :  
`SELECT * FROM table`
- Pour récupérer simplement les champs `champ1`, `champ2`, ... on utilise :  
`SELECT champ1, champ2, ... FROM table`

## Exemples

- `SELECT City, Oyear FROM Medals` renvoie une table à deux colonnes : les villes olympiques et l'année.

## Premiers pas en SQL

- Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :  
`SELECT * FROM table`
- Pour récupérer simplement les champs `champ1`, `champ2`, ... on utilise :  
`SELECT champ1, champ2, ... FROM table`

## Exemples

- `SELECT City, Oyear FROM Medals` renvoie une table à deux colonnes : les villes olympiques et l'année.  
! Pour chaque enregistrement on affiche la ville et l'année donc on obtient des répétitions :

City	Oyear
Montreal	1976
Montreal	1976
...	...

### Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

### Exemples

### Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : =, <, >, <=, >=, <> (différent) et **BETWEEN** (entre)

### Exemples

### Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **BETWEEN** (entre)
- Logique : **and**, **or** et **not**

### Exemples

## Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **BETWEEN** (entre)
- Logique : **and**, **or** et **not**
- Modèle de chaînes de caractères : **LIKE** où **%** désigne n'importe quel suite de caractères et **\_** un unique caractère

## Exemples



## Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **BETWEEN** (entre)
- Logique : **and**, **or** et **not**
- Modèle de chaînes de caractères : **LIKE** où **%** désigne n'importe quel suite de caractères et **\_** un unique caractère

## Exemples

Pour chercher dans la table les champions olympiques français des JO de 1980

## Clause WHERE

Une instruction **SELECT** peut être suivie d'une clause **WHERE** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **BETWEEN** (entre)
- Logique : **and**, **or** et **not**
- Modèle de chaînes de caractères : **LIKE** où **%** désigne n'importe quel suite de caractères et **\_** un unique caractère

## Exemples

Pour chercher dans la table les champions olympiques français des JO de 1980

```
SELECT Athlete FROM Medals WHERE Country="France" AND Medal="Gold" AND Oyear="1980"
```

### Clause ORDER BY

Une instruction **SELECT** peut être suivie d'une clause **ORDER BY** qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

### Exemples

## Clause ORDER BY

Une instruction `SELECT` peut être suivie d'une clause `ORDER BY` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

- `ASC` pour indiquer un classement par ordre croissant

## Exemples

## Clause ORDER BY

Une instruction **SELECT** peut être suivie d'une clause **ORDER BY** qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

- **ASC** pour indiquer un classement par ordre croissant
- **DESC** pour indiquer un classement par ordre décroissant

## Exemples

### Clause ORDER BY

Une instruction **SELECT** peut être suivie d'une clause **ORDER BY** qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

- **ASC** pour indiquer un classement par ordre croissant
- **DESC** pour indiquer un classement par ordre décroissant

La valeur par défaut est **ASC**

### Exemples

### Clause ORDER BY

Une instruction **SELECT** peut être suivie d'une clause **ORDER BY** qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

- **ASC** pour indiquer un classement par ordre croissant
- **DESC** pour indiquer un classement par ordre décroissant

La valeur par défaut est **ASC**

### Exemples

- Pour classer par ordre alphabétique les noms vainqueurs du 100 m aux JO :

### Clause ORDER BY

Une instruction **SELECT** peut être suivie d'une clause **ORDER BY** qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle-même suivie de :

- **ASC** pour indiquer un classement par ordre croissant
- **DESC** pour indiquer un classement par ordre décroissant

La valeur par défaut est **ASC**

### Exemples

- Pour classer par ordre alphabétique les noms vainqueurs du 100 m aux JO :  

```
SELECT Athlete FROM Medals WHERE Event="100m" AND Medal="Gold" ORDER BY Athlete
```



### Clause DISTINCT et LIMIT

### Exemples

### Clause DISTINCT et LIMIT

- Une instruction `SELECT` peut être *directement* suivie d'une clause `DISTINCT` champ qui indique que champ ne doit apparaître qu'une fois dans les résultats

### Exemples

## Clause DISTINCT et LIMIT

- Une instruction **SELECT** peut être *directement* suivie d'une clause **DISTINCT** champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction **SELECT** peut être suivie d'une clause **LIMIT** qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec **ORDER BY**.

## Exemples

## Clause DISTINCT et LIMIT

- Une instruction **SELECT** peut être *directement* suivie d'une clause **DISTINCT** champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction **SELECT** peut être suivie d'une clause **LIMIT** qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec **ORDER BY**.

## Exemples

### Clause DISTINCT et LIMIT

- Une instruction **SELECT** peut être *directement* suivie d'une clause **DISTINCT** champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction **SELECT** peut être suivie d'une clause **LIMIT** qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec **ORDER BY**.

### Exemples

- Pour afficher les villes olympiques sans répétitions :

## Clause DISTINCT et LIMIT

- Une instruction `SELECT` peut être *directement* suivie d'une clause `DISTINCT` champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction `SELECT` peut être suivie d'une clause `LIMIT` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `ORDER BY`.

## Exemples

- Pour afficher les villes olympiques sans répétitions :  
`SELECT DISTINCT City FROM Medals`

## Clause DISTINCT et LIMIT

- Une instruction `SELECT` peut être *directement* suivie d'une clause `DISTINCT` champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction `SELECT` peut être suivie d'une clause `LIMIT` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `ORDER BY`.

## Exemples

- Pour afficher les villes olympiques sans répétitions :  
`SELECT DISTINCT City FROM Medals`
- Pour afficher les trois derniers champions olympiques du décathlon

### Clause DISTINCT et LIMIT

- Une instruction **SELECT** peut être *directement* suivie d'une clause **DISTINCT** champ qui indique que champ ne doit apparaître qu'une fois dans les résultats
- Une instruction **SELECT** peut être suivie d'une clause **LIMIT** qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec **ORDER BY**.

### Exemples

- Pour afficher les villes olympiques sans répétitions :  
`SELECT DISTINCT City FROM Medals`
- Pour afficher les trois derniers champions olympiques du décathlon  
`SELECT Athlete FROM Medals WHERE Event="decathlon" AND Medal="Gold"  
ORDER BY OYear DESC LIMIT 3`



### Renommage

- On peut faire des calculs dans les requêtes

### Exemples

### Renommage

- On peut faire des calculs dans les requêtes
- On peut renommer une colonne avec `AS`

### Exemples

### Renommage

- On peut faire des calculs dans les requêtes
- On peut renommer une colonne avec `AS`
- Cela est particulièrement utile pour y faire référence ensuite

### Exemples

### Renommage

- On peut faire des calculs dans les requêtes
- On peut renommer une colonne avec **AS**
- Cela est particulièrement utile pour y faire référence ensuite

### Exemples

On calcule l'ancienneté de chaque ville olympique et on les affiche par ordre croissant.

## Renommage

- On peut faire des calculs dans les requêtes
- On peut renommer une colonne avec `AS`
- Cela est particulièrement utile pour y faire référence ensuite

## Exemples

On calcule l'ancienneté de chaque ville olympique et on les affiche par ordre croissant. `SELECT DISTINCT City, 2023-0year AS Ancienneté FROM Medals ORDER BY Ancienneté DESC`

### Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

### Exemples

## Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

## Exemples

### Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max

### Exemples



### Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max
- SUM pour obtenir la somme

### Exemples

## Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max
- SUM pour obtenir la somme
- AVG pour obtenir le minimum

## Exemples

### Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max
- SUM pour obtenir la somme
- AVG pour obtenir le minimum
- COUNT pour compter le nombre d'enregistrement

### Exemples

## Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max
- SUM pour obtenir la somme
- AVG pour obtenir le minimum
- COUNT pour compter le nombre d'enregistrement

## Exemples

- Pour compter le nombre de médailles de bronze Française en 2008 :

### Agrégation

Le langage SQL offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- MIN pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)
- MAX pour obtenir le max
- SUM pour obtenir la somme
- AVG pour obtenir le minimum
- COUNT pour compter le nombre d'enregistrement

### Exemples

- Pour compter le nombre de médailles de bronze Française en 2008 :  

```
SELECT COUNT(*) FROM Medals WHERE Medal="Bronze" AND  
Country="France" AND Oyear=2008
```

### Clause GROUP BY

- On peut regrouper les résultats pour un attribut donné à l'aide de **GROUP BY**.

### Exemples

### Clause GROUP BY

- On peut regrouper les résultats pour un attribut donné à l'aide de **GROUP BY**.
- Un seul résultat sera affiché pour chaque valeur possible de l'attribut.

### Exemples

### Clause GROUP BY

- On peut regrouper les résultats pour un attribut donné à l'aide de **GROUP BY**.
- Un seul résultat sera affiché pour chaque valeur possible de l'attribut.
- Les fonctions d'agrégation dans le **SELECT** s'appliquent alors à chaque groupe.

### Exemples



### Clause GROUP BY

- On peut regrouper les résultats pour un attribut donné à l'aide de **GROUP BY**.
- Un seul résultat sera affiché pour chaque valeur possible de l'attribut.
- Les fonctions d'agrégation dans le **SELECT** s'appliquent alors à chaque groupe.

### Exemples

Pour afficher le nombre total de médailles par pays

### Clause GROUP BY

- On peut regrouper les résultats pour un attribut donné à l'aide de **GROUP BY**.
- Un seul résultat sera affiché pour chaque valeur possible de l'attribut.
- Les fonctions d'agrégation dans le **SELECT** s'appliquent alors à chaque groupe.

### Exemples

Pour afficher le nombre total de médailles par pays


```
SELECT Country, COUNT(*) AS total FROM Medals GROUP BY Country
```

### Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.


### Exemples

### Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.
-  Ne pas confondre :

### Exemples

### Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.
-  Ne pas confondre :
  - `WHERE` qui donne une condition sur les *enregistrements* à afficher.

### Exemples

### Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.
- **!** Ne pas confondre :
  - `WHERE` qui donne une condition sur les *enregistrements* à afficher.
  - `HAVING` qui est utilisé à la suite de `GROUP BY` pour donner une condition sur les groupes à afficher.

### Exemples


## Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.
- **!** Ne pas confondre :
  - `WHERE` qui donne une condition sur les *enregistrements* à afficher.
  - `HAVING` qui est utilisé à la suite de `GROUP BY` pour donner une condition sur les groupes à afficher.

## Exemples

Pour afficher les pays ayant eu au total plus de 100 médailles

### Clause HAVING

- Une clause `GROUP BY` peut être complétée par une clause `HAVING` qui indique une condition sur les groupes à afficher.
-  Ne pas confondre :
  - `WHERE` qui donne une condition sur les *enregistrements* à afficher.
  - `HAVING` qui est utilisé à la suite de `GROUP BY` pour donner une condition sur les groupes à afficher.

### Exemples

Pour afficher les pays ayant eu au total plus de 100 médailles

```
SELECT Country, COUNT(*) AS total FROM Medals GROUP BY Country HAVING total>100
```