

Définitions

- Le **modèle relationnel**, introduit en 1969 par E. CODD est une réponse à la problématique de la gestion des données en informatique :
 - Accès concurrents aux mêmes données par des utilisateurs différents

Définitions

- Le **modèle relationnel**, introduit en 1969 par E. CODD est une réponse à la problématique de la gestion des données en informatique :
 - Accès concurrents aux mêmes données par des utilisateurs différents
 - Indépendance logique

Définitions

- Le **modèle relationnel**, introduit en 1969 par E. CODD est une réponse à la problématique de la gestion des données en informatique :
 - Accès concurrents aux mêmes données par des utilisateurs différents
 - Indépendance logique
 - Résistance aux pannes

Définitions

- Le **modèle relationnel**, introduit en 1969 par E. CODD est une réponse à la problématique de la gestion des données en informatique :
 - Accès concurrents aux mêmes données par des utilisateurs différents
 - Indépendance logique
 - Résistance aux pannes
- Dans le modèle relationnel, une **base de données** est un ensemble de **tables** (appelés aussi **relations**).

Définitions

- Le **modèle relationnel**, introduit en 1969 par E. CODD est une réponse à la problématique de la gestion des données en informatique :
 - Accès concurrents aux mêmes données par des utilisateurs différents
 - Indépendance logique
 - Résistance aux pannes
- Dans le modèle relationnel, une **base de données** est un ensemble de **tables** (appelés aussi **relations**).
- Un **SGBD** (Système de Gestion de Base de Données), en anglais **DBMS**, (DataBase Management Service) est un logiciel permettant de gérer et d'interagir avec une base de données. Les plus connus sont MySQL (libre) ou Oracle (propriétaire).

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).
- Un attribut est caractérisé par son **nom** et son **domaine** (ensemble dans lequel il peut prendre ses valeurs).

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).
- Un attribut est caractérisé par son **nom** et son **domaine** (ensemble dans lequel il peut prendre ses valeurs).
- Un n-uplet (donc une ligne dans une relation) s'appelle un **enregistrement**, il modélise une entité du monde réel.

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).
- Un attribut est caractérisé par son **nom** et son **domaine** (ensemble dans lequel il peut prendre ses valeurs).
- Un n-uplet (donc une ligne dans une relation) s'appelle un **enregistrement**, il modélise une entité du monde réel.
- Une **clé primaire** est un attribut (ou ensemble d'attribut) permettant d'identifier de façon unique un enregistrement.

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).
- Un attribut est caractérisé par son **nom** et son **domaine** (ensemble dans lequel il peut prendre ses valeurs).
- Un n-uplet (donc une ligne dans une relation) s'appelle un **enregistrement**, il modélise une entité du monde réel.
- Une **clé primaire** est un attribut (ou ensemble d'attribut) permettant d'identifier de façon unique un enregistrement.
- Le **schéma** d'une table consiste en la donnée de son nom et des noms des attributs avec le domaine de chacun.

Modèle relationnel

- Une **relation** est un **ensemble** (donc pas de redondance) de n-uplets, tous de même taille.
- Chaque composante d'un n-uplet est appelé **attribut** (ou champ).
- Un attribut est caractérisé par son **nom** et son **domaine** (ensemble dans lequel il peut prendre ses valeurs).
- Un n-uplet (donc une ligne dans une relation) s'appelle un **enregistrement**, il modélise une entité du monde réel.
- Une **clé primaire** est un attribut (ou ensemble d'attribut) permettant d'identifier de façon unique un enregistrement.
- Le **schéma** d'une table consiste en la donnée de son nom et des noms des attributs avec le domaine de chacun.
- Le **schéma relationnel** d'une base de données est l'ensemble des schémas des tables de cette base.

Un exemple de relation

Id	Langage	Année	Inventeur(s)
1	C	1972	Dennis Ritchie
2	C++	1983	Bjarne Stroustrup
3	Java	1995	James Gosling
4	Python	1991	Guido van Rossum
5	JavaScript	1995	Brendan Eich
6	PHP	1995	Rasmus Lerdorf
7	C#	2000	Microsoft
8	Swift	2014	Apple
9	Go	2009	Google
10	Rust	2010	Mozilla
11	OCaml	1996	Xavier Leroy et al.

Un exemple de relation

Id	Langage	Année	Inventeur(s)
1	C	1972	Dennis Ritchie
2	C++	1983	Bjarne Stroustrup
3	Java	1995	James Gosling
4	Python	1991	Guido van Rossum
5	JavaScript	1995	Brendan Eich
6	PHP	1995	Rasmus Lerdorf
7	C#	2000	Microsoft
8	Swift	2014	Apple
9	Go	2009	Google
10	Rust	2010	Mozilla
11	OCaml	1996	Xavier Leroy et al.

→ Un enregistrement

Un exemple de relation

Id	Langage	Année	Inventeur(s)
1	C	1972	Dennis Ritchie
2	C++	1983	Bjarne Stroustrup
3	Java	1995	James Gosling
4	Python	1991	Guido van Rossum
5	JavaScript	1995	Brendan Eich
6	PHP	1995	Rasmus Lerdorf
7	C#	2000	Microsoft
8	Swift	2014	Apple
9	Go	2009	Google
10	Rust	2010	Mozilla
11	OCaml	1996	Xavier Leroy et al.

→ L'**Attribut** de nom *Année* de domaine \mathbb{N} .

Un exemple de relation

Id peut servir de clé primaire car unique pour chaque enregistrement

Id	Langage	Année	Inventeur(s)
1	C	1972	Dennis Ritchie
2	C++	1983	Bjarne Stroustrup
3	Java	1995	James Gosling
4	Python	1991	Guido van Rossum
5	JavaScript	1995	Brendan Eich
6	PHP	1995	Rasmus Lerdorf
7	C#	2000	Microsoft
8	Swift	2014	Apple
9	Go	2009	Google
10	Rust	2010	Mozilla
11	OCaml	1996	Xavier Leroy et al.

C11 Base de données, modèle relationnel

2. Schéma relationnel d'une base de données

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table des langages de programmation peut s'écrire :

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table des langages de programmation peut s'écrire :

- Sous forme de liste :

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table des langages de programmation peut s'écrire :

- Sous forme de liste :

langages (id : INT, Langage : TEXT, Année : INT, Inventeurs : TEXT)

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table des langages de programmation peut s'écrire :

- Sous forme de liste :

langages (id : INT, Langage : TEXT, Année : INT, Inventeurs : TEXT)

- Sous forme de tableau :

personnes	
<u>id</u>	: INT
Langage	: TEXT
Année	: INT
Inventeurs	: TEXT

Duplication de l'information

- On évite pour de multiples raisons (espace occupé, efficacité pour les recherches ou les modifications, ...) de dupliquer l'information présente dans une base de données.

Duplication de l'information

- On évite pour de multiples raisons (espace occupé, efficacité pour les recherches ou les modifications, ...) de dupliquer l'information présente dans une base de données.
- On est donc amené à créer plusieurs tables **liées** entre elles, c'est à dire que certains attributs d'une table sont les clés primaires d'une autre table. On dit que ce sont des **clés étrangères**.

Duplication de l'information

- On évite pour de multiples raisons (espace occupé, efficacité pour les recherches ou les modifications, ...) de dupliquer l'information présente dans une base de données.
- On est donc amené à créer plusieurs tables **liées** entre elles, c'est à dire que certains attributs d'une table sont les clés primaires d'une autre table. On dit que ce sont des **clés étrangères**.
- Les clés étrangères sont précédées de # dans le schéma relationnel.

Exemple

Id	Chanson	Interprète	Nationalité	Naissance	Année de Sortie	Album
1	"Bohemian Rhapsody"	Queen	Britannique	1946	1975	"A Night at the Opera"
2	"Like a Rolling Stone"	Bob Dylan	Américain	1941	1965	"Highway 61 Revisited"
3	"Imagine"	John Lennon	Britannique	1940	1971	"Imagine"
4	"Billie Jean"	Michael Jackson	Américain	1958	1983	"Thriller"
5	"Amstrong"	Claude Nougaro	Français	1929	1984	"Nougayork"

Exemple

Id	Chanson	Interprète	Nationalité	Naissance	Année de Sortie	Album
1	"Bohemian Rhapsody"	Queen	Britannique	1946	1975	"A Night at the Opera"
2	"Like a Rolling Stone"	Bob Dylan	Américain	1941	1965	"Highway 61 Revisited"
3	"Imagine"	John Lennon	Britannique	1940	1971	"Imagine"
4	"Billie Jean"	Michael Jackson	Américain	1958	1983	"Thriller"
5	"Amstrong"	Claude Nougaro	Français	1929	1984	"Nougayork"

- Donner le schéma relationnel de cet table

Exemple

Id	Chanson	Interprète	Nationalité	Naissance	Année de Sortie	Album
1	"Bohemian Rhapsody"	Queen	Britannique	1946	1975	"A Night at the Opera"
2	"Like a Rolling Stone"	Bob Dylan	Américain	1941	1965	"Highway 61 Revisited"
3	"Imagine"	John Lennon	Britannique	1940	1971	"Imagine"
4	"Billie Jean"	Michael Jackson	Américain	1958	1983	"Thriller"
5	"Amstrong"	Claude Nougaro	Français	1929	1984	"Nougayork"

- Donner le schéma relationnel de cet table
- Proposer un schéma composé de plusieurs tables et permettant de représenter plus efficacement les données.

Exemple

Exemple

On souhaite modéliser un relevé de notes sur lequel figure :

Exemple

On souhaite modéliser un relevé de notes sur lequel figure :

- Un élève (nom, prénom, date de naissance, et identifiant unique)

Exemple

On souhaite modéliser un relevé de notes sur lequel figure :

- Un élève (nom, prénom, date de naissance, et identifiant unique)
- Un ensemble de matière fixées

Exemple

On souhaite modéliser un relevé de notes sur lequel figure :

- Un élève (nom, prénom, date de naissance, et identifiant unique)
- Un ensemble de matière fixées
- Au plus une note par matière et par élève

Exemple

On souhaite modéliser un relevé de notes sur lequel figure :

- Un élève (nom, prénom, date de naissance, et identifiant unique)
- Un ensemble de matière fixées
- Au plus une note par matière et par élève

Expliquer pourquoi un schéma relationnel d'une seule table notes n'est pas satisfaisant et proposer un schéma relationnel constitué de 3 tables

Intégrité référentielle

Dans une base de données, l'**intégrité référentielle**, assure qu'une clé étrangère est toujours présente en tant que clé primaire dans la table qu'elle référence. Cela assure la cohérence des données et empêche des suppressions par erreur de la part des utilisateurs.

Exemple

Intégrité référentielle

Dans une base de données, l'**intégrité référentielle**, assure qu'une clé étrangère est toujours présente en tant que clé primaire dans la table qu'elle référence. Cela assure la cohérence des données et empêche des suppressions par erreur de la part des utilisateurs.

Exemple

Dans l'exemple précédent, la table des notes doit toujours faire référence à un élève et à une matière. L'intégrité référentielle préserve automatiquement la relation "une note appartient forcément à un élève" et "une note est donnée dans une matière existante"

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.

Généralités

- SQL (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.

Généralités

- SQL (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.
- En MP2I seules les requêtes permettant d'extraire des informations d'une ou plusieurs tables sont étudiées. Toutes nos requêtes seront donc basées sur l'instruction `SELECT`.

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.
- En MP2I seules les requêtes permettant d'extraire des informations d'une ou plusieurs tables sont étudiées. Toutes nos requêtes seront donc basées sur l'instruction `SELECT`.
- On se limitera aux domaines (types) suivant :

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.
- En MP2I seules les requêtes permettant d'extraire des informations d'une ou plusieurs tables sont étudiées. Toutes nos requêtes seront donc basées sur l'instruction `SELECT`.
- On se limitera aux domaines (types) suivant :
 - `STRING` : chaînes de caractères

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.
- En MP2I seules les requêtes permettant d'extraire des informations d'une ou plusieurs tables sont étudiées. Toutes nos requêtes seront donc basées sur l'instruction `SELECT`.
- On se limitera aux domaines (types) suivant :
 - `STRING` : chaînes de caractères
 - `INT` : nombres entiers

Généralités

- **SQL** (*Structured Query Language*) est un langage de requête permettant d'interagir avec une base de données et d'y récupérer des informations.
- Le standard du langage n'est pas parfaitement mis en oeuvre et des différences notables peuvent exister entre différents SGBD.
- Le langage permet de créer, modifier, mettre à jour, ... les tables d'une base de données.
- En MP2I seules les requêtes permettant d'extraire des informations d'une ou plusieurs tables sont étudiées. Toutes nos requêtes seront donc basées sur l'instruction `SELECT`.
- On se limitera aux domaines (types) suivant :
 - `STRING` : chaînes de caractères
 - `INT` : nombres entiers
 - `FLOAT` : nombres en virgule flottante

Exemple

Pour illustrer le cours on utilise une table des médailles obtenus au JO :

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

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, LIMIT et OFFSET

Exemples

Clause DISTINCT, LIMIT et OFFSET

- 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, LIMIT et OFFSET

- 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, LIMIT et OFFSET

- 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**.
- **OFFSET** permet de se décaler dans l'ordre des résultats.

Exemples

Clause DISTINCT, LIMIT et OFFSET

- 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**.
- **OFFSET** permet de se décaler dans l'ordre des résultats.

Exemples

- Pour afficher les villes olympiques sans répétitions :

Clause DISTINCT, LIMIT et OFFSET

- 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**.
- **OFFSET** permet de se décaler dans l'ordre des résultats.

Exemples

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

Clause DISTINCT, LIMIT et OFFSET

- 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**.
- **OFFSET** permet de se décaler dans l'ordre des résultats.

Exemples

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

4. Requêtes sur une seule table

Clause DISTINCT, LIMIT et OFFSET

- 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**.
- **OFFSET** permet de se décaler dans l'ordre des résultats.

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
```

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions
- Lister pr ordre croissant les cinq pays ayant le plus d'habitant

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions
- Lister pr ordre croissant les cinq pays ayant le plus d'habitant
- Donner la surface de la France

4. Requêtes sur une seule table

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions
- Lister pr ordre croissant les cinq pays ayant le plus d'habitant
- Donner la surface de la France
- Lister les pays ayant la densité de population la plus faible (rapport entre le nombre d'habitants et la surface du pays)

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions
- Lister pr ordre croissant les cinq pays ayant le plus d'habitant
- Donner la surface de la France
- Lister les pays ayant la densité de population la plus faible (rapport entre le nombre d'habitants et la surface du pays)
- Donner le plus grand pays n'ayant pas d'ouverture sur la mer.

Exemple

On considère une base de données des pays du monde constitué d'une seule table : **Pays** (Id : INTEGER, Country : TEXT, Region : TEXT, Population : INTEGER, Area : INTEGER, Coastline : FLOAT, GDP : INTEGER)

Voici un exemple d'enregistrement dans cette table :

(20, "Belgium", "WESTERN EUROPE", 10379067, 30528, 0.22, 29100)

Ecrire les requêtes permettant :

- Lister les différentes régions
- Lister pr ordre croissant les cinq pays ayant le plus d'habitant
- Donner la surface de la France
- Lister les pays ayant la densité de population la plus faible (rapport entre le nombre d'habitants et la surface du pays)
- Donner le plus grand pays n'ayant pas d'ouverture sur la mer.
- Donner la superficie moyenne des pays en les regroupant par région

Union de deux tables

Lorsque deux tables T_1 et T_2 ont **le même schéma relationnel** (c'est à dire les même colonnes), $T_1 \cup T_2$ contient les enregistrement de T_1 ou T_2 (sans duplication). La syntaxe SQL correspondante est :

Exemple

Union de deux tables

Lorsque deux tables T_1 et T_2 ont **le même schéma relationnel** (c'est à dire les même colonnes), $T_1 \cup T_2$ contient les enregistrement de T_1 ou T_2 (sans duplication). La syntaxe SQL correspondante est :

```
SELECT * FROM T1 UNION SELECT * FROM T2;
```

Exemple

Union de deux tables

Lorsque deux tables T_1 et T_2 ont le même schéma relationnel (c'est à dire les mêmes colonnes), $T_1 \cup T_2$ contient les enregistrements de T_1 ou T_2 (sans duplication). La syntaxe SQL correspondante est :

```
SELECT * FROM T1 UNION SELECT * FROM T2;
```

Exemple

Table T_1

Id	Nom	Prénom
7	Payet	Jean
28	Hoarau	Paul
42	Untel	Sam
57	Casimir	Tom

Table T_2

Id	Nom	Prénom
12	Martin	Pierre
42	Untel	Sam
45	Grondin	Eric

Table $T_1 \cup T_2$

Id	Nom	Prénom
7	Payet	Jean
28	Hoarau	Paul
42	Untel	Sam
57	Casimir	Tom
12	Martin	Pierre
45	Grondin	Eric

Intersection de deux tables

Lorsque deux tables T_1 et T_2 ont **le même schéma relationnel** (c'est à dire les même colonnes), $T_1 \cap T_2$ contient les enregistrement apparaissant dans T_1 et dans T_2 .

Exemple

Intersection de deux tables

Lorsque deux tables T_1 et T_2 ont le même schéma relationnel (c'est à dire les mêmes colonnes), $T_1 \cap T_2$ contient les enregistrements apparaissant dans T_1 et dans T_2 . La syntaxe SQL correspondante est :

```
SELECT * FROM T1 INTERSECT SELECT * FROM T2 ;
```

Exemple

Intersection de deux tables

Lorsque deux tables T_1 et T_2 ont le même schéma relationnel (c'est à dire les mêmes colonnes), $T_1 \cap T_2$ contient les enregistrement apparaissant dans T_1 et dans T_2 . La syntaxe SQL correspondante est :

```
SELECT * FROM T1 INTERSECT SELECT * FROM T2 ;
```

Exemple

Table T_1

Id	Nom	Prénom
7	Payet	Jean
28	Hoarau	Paul
42	Untel	Sam
57	Casimir	Tom

Table T_2

Id	Nom	Prénom
12	Martin	Pierre
42	Untel	Sam
45	Grondin	Eric

Table $T_1 \cap T_2$

Id	Nom	Prénom
42	Untel	Sam

Différence de deux tables

Lorsque deux tables T_1 et T_2 ont **le même schéma relationnel** (c'est à dire les même colonnes), $T_1 - T_2$ contient les enregistrement apparaissant dans T_1 et pas dans T_2 .

Exemple

Différence de deux tables

Lorsque deux tables T_1 et T_2 ont le même schéma relationnel (c'est à dire les même colonnes), $T_1 - T_2$ contient les enregistrement apparaissant dans T_1 et pas dans T_2 . La syntaxe SQL correspondante est :

```
SELECT * FROM T1 EXCEPT SELECT * FROM T2 ;
```

Exemple

Différence de deux tables

Lorsque deux tables T_1 et T_2 ont le même schéma relationnel (c'est à dire les mêmes colonnes), $T_1 - T_2$ contient les enregistrement apparaissant dans T_1 et pas dans T_2 . La syntaxe SQL correspondante est :

```
SELECT * FROM T1 EXCEPT SELECT * FROM T2 ;
```

Exemple

Table T_1

Id	Nom	Prénom
7	Payet	Jean
28	Hoarau	Paul
42	Untel	Sam
57	Casimir	Tom

Table T_2

Id	Nom	Prénom
12	Martin	Pierre
42	Untel	Sam
45	Grondin	Eric

Table $T_1 - T_2$

Id	Nom	Prénom
7	Payet	Jean
28	Hoarau	Paul
57	Casimir	Tom

Produit cartésien de deux tables

On peut réaliser le **produit cartésien** de deux tables, c'est à dire l'ensemble des enregistrements formé d'un enregistrement de la première table et d'un enregistrement de la seconde.

Exemple

Produit cartésien de deux tables

On peut réaliser le **produit cartésien** de deux tables, c'est à dire l'ensemble des enregistrements formé d'un enregistrement de la première table et d'un enregistrement de la seconde.

La syntaxe SQL correspondante est :

```
SELECT * FROM T1, T2 ;
```

Exemple

C11 Base de données, modèle relationnel

5. Requêtes sur plusieurs tables : opérations ensemblistes

Produit cartésien de deux tables

On peut réaliser le **produit cartésien** de deux tables, c'est à dire l'ensemble des enregistrements formé d'un enregistrement de la première table et d'un enregistrement de la seconde.

La syntaxe SQL correspondante est :

```
SELECT * FROM T1, T2 ;
```

Exemple

Table T_1

Id	Nom	Prénom
7	Payet	Jean
42	Untel	Sam

Table T_2

Matière	Note
Info	15
Maths	9
Physique	10

Table $T_1 \times T_2$

Id	Nom	Prénom	Matière	Note
7	Payet	Jean	Info	15
42	Untel	Sam	Info	15
7	Payet	Jean	Maths	9
42	Untel	Sam	Maths	9
7	Payet	Jean	Physique	10
42	Untel	Sam	Physique	10

Définition

- La jointure de deux tables T_1 et T_2 sur les colonnes A et B revient à combiner les enregistrements de T_1 et T_2 lorsque les colonnes A et B coïncident.

Définition

- La jointure de deux tables T_1 et T_2 sur les colonnes A et B revient à combiner les enregistrements de T_1 et T_2 lorsque les colonnes A et B coïncident.
- Cette jointure se note $T_1 \bowtie_{A=B} T_2$

Définition

- La jointure de deux tables T_1 et T_2 sur les colonnes A et B revient à combiner les enregistrements de T_1 et T_2 lorsque les colonnes A et B coïncident.
- Cette jointure se note $T_1 \bowtie_{A=B} T_2$
- Cette notion est **fortement liée** à celle de clé étrangère, on effectuera souvent les jointures avec A une clé primaire et B une clé étrangère correspondante.

Définition

- La jointure de deux tables T_1 et T_2 sur les colonnes A et B revient à combiner les enregistrements de T_1 et T_2 lorsque les colonnes A et B coïncident.
- Cette jointure se note $T_1 \bowtie_{A=B} T_2$
- Cette notion est **fortement liée** à celle de clé étrangère, on effectuera souvent les jointures avec A une clé primaire et B une clé étrangère correspondante.
- La syntaxe SQL correspondante est :
SELECT * FROM T1 JOIN T2 on T1.A = T2.B

Définition

- La jointure de deux tables T_1 et T_2 sur les colonnes A et B revient à combiner les enregistrements de T_1 et T_2 lorsque les colonnes A et B coïncident.
- Cette jointure se note $T_1 \bowtie_{A=B} T_2$
- Cette notion est **fortement liée** à celle de clé étrangère, on effectuera souvent les jointures avec A une clé primaire et B une clé étrangère correspondante.
- La syntaxe SQL correspondante est :
`SELECT * FROM T1 JOIN T2 on T1.A = T2.B`
- On peut joindre plus de deux tables.

Exemple

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Table Livres

Num	Auteur	Titre
1	4	Dune
2	1	Les robots
3	7	L'île mystérieuse
4	1	Fondation

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Table Livres

Num	Auteur	Titre
1	4	Dune
2	1	Les robots
3	7	L'île mystérieuse
4	1	Fondation

La jointure de **Auteurs** et **Livres** sur les colonnes Id et Auteur donne :

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Table Livres

Num	Auteur	Titre
1	4	Dune
2	1	Les robots
3	7	L'île mystérieuse
4	1	Fondation

La jointure de **Auteurs** et **Livres** sur les colonnes **Id** et **Auteur** donne :

Id	Prénom	Nom	Num	Auteur	Titre
1	Isaac	Asimov	2	1	Les robots
1	Isaac	Asimov	4	1	Fondation
4	Franck	Herbert	1	4	Dune
7	Jules	Verne	3	7	L'île mystérieuse

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Table Livres

Num	Auteur	Titre
1	4	Dune
2	1	Les robots
3	7	L'île mystérieuse
4	1	Fondation

La jointure de **Auteurs** et **Livres** sur les colonnes **Id** et **Auteur** donne :

Id	Prénom	Nom	Num	Auteur	Titre
1	Isaac	Asimov	2	1	Les robots
1	Isaac	Asimov	4	1	Fondation
4	Franck	Herbert	1	4	Dune
7	Jules	Verne	3	7	L'île mystérieuse

```
SELECT * FROM Auteurs JOIN Livres on Auteurs.ID = Livres.Auteur
```

Exemple

Table Auteurs

Id	Prénom	Nom
1	Isaac	Asimov
4	Franck	Herbert
7	Jules	Verne

Table Livres

Num	Auteur	Titre
1	4	Dune
2	1	Les robots
3	7	L'île mystérieuse
4	1	Fondation

La jointure de **Auteurs** et **Livres** sur les colonnes **Id** et **Auteur** donne :

Id	Prénom	Nom	Num	Auteur	Titre
1	Isaac	Asimov	2	1	Les robots
1	Isaac	Asimov	4	1	Fondation
4	Franck	Herbert	1	4	Dune
7	Jules	Verne	3	7	L'île mystérieuse

SELECT * FROM Auteurs JOIN Livres on Auteurs.ID = Livres.Auteur
(On préfixe le nom des attributs par celui de la table afin d'éviter toute ambiguïté.)

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.

Exemple

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.

Exemple

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.
- C'est souvent dans un **WHERE** ou un **HAVING**

Exemple

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.
- C'est souvent dans un **WHERE** ou un **HAVING**

Exemple

Dans la relation *Objet* (Référence : INT, description : TEXT, prix : FLOAT), comment retrouver le (ou les) objets ayant le prix le plus élevé ?

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.
- C'est souvent dans un **WHERE** ou un **HAVING**

Exemple

Dans la relation *Objet* (Référence : INT, description : TEXT, prix : FLOAT), comment retrouver le (ou les) objets ayant le prix le plus élevé ?

- **SELECT** Référence, **MAX**(prix) **FROM** objet;
Cette solution n'est pas satisfaisante car elle renverra une seule référence même si plusieurs objets ont le prix maximal.

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.
- C'est souvent dans un **WHERE** ou un **HAVING**

Exemple

Dans la relation *Objet* (Référence : INT, description : TEXT, prix : FLOAT), comment retrouver le (ou les) objets ayant le prix le plus élevé ?

- **SELECT** Référence, **MAX**(prix) **FROM** objet;
Cette solution n'est pas satisfaisante car elle renverra une seule référence même si plusieurs objets ont le prix maximal.
- **SELECT** Référence, prix **FROM** objet **ORDER BY** prix **DESC LIMIT** 1;
De même pour cette solution !

Requête dans le résultat d'une requête

- Le résultat d'une requête peut-être utilisé afin d'effectuer une autre requête.
- C'est le principe des **requêtes imbriquées**.
- C'est souvent dans un **WHERE** ou un **HAVING**

Exemple

Dans la relation *Objet* (Référence : INT, description : TEXT, prix : FLOAT), comment retrouver le (ou les) objets ayant le prix le plus élevé ?

- **SELECT** Référence, **MAX**(prix) **FROM** objet;
Cette solution n'est pas satisfaisante car elle renverra une seule référence même si plusieurs objets ont le prix maximal.
- **SELECT** Référence, prix **FROM** objet **ORDER BY** prix **DESC LIMIT** 1;
De même pour cette solution !
- **SELECT** Référence, prix **FROM** Objet
WHERE note = (**SELECT** **MAX**(prix) **FROM** objet);