Bases de données relationnelles

Jean-Pierre Becirspahic
Lycée Louis-Le-Grand

Classification décimale de Dewey

Inventée au XIX^{e} siècle, elle range les ouvrages en 10 divisions.

- 000 Informatique, information, ouvrages généraux ;
- 100 Philosophie, Parapsychologie et Occultisme, Psychologie ;
- 200 Religion ;
- 300 Sciences sociales ;
- 400 Langues ;
- 500 Sciences de la nature et Mathématiques ;
- 600 Technologie (Sciences appliquées) ;
- 700 Arts, Loisirs et Sports ;
- 800 Littérature (Belles-Lettres) et techniques d'écriture ;
- 900 Géographie, Histoire et disciplines auxiliaires.

Classification décimale de Dewey

Inventée au XIX^{e} siècle, elle range les ouvrages en 10 divisions.

Chaque division est rangée en 10 subdivisions.

- 500 Généralités sur les sciences de la nature et les mathématiques ;
- 510 Mathématiques ;
- 520 Astronomie et sciences connexes ;
- 530 Physique ;
- 540 Chimie et sciences connexes ;
- 550 Sciences de la Terre ;
- 560 Paléontologie, Paléozoologie ;
- 570 Sciences de la vie, Biologie ;
- 580 Plantes, Botanique ;
- 590 Animaux, Zoologie.

Classification décimale de Dewey

Inventée au XIX^{e} siècle, elle range les ouvrages en 10 divisions.

Chaque division est rangée en 10 subdivisions.

Chaque subdivision est elle-même subdivisée.

500 Sciences de la nature et Mathématiques ;

530 Physique ;

537 Électricité et électronique ;

537.2 Électrostatique ;

Classification décimale de Dewey

Inventée au XIX^{e} siècle, elle range les ouvrages en 10 divisions.

Chaque division est rangée en 10 subdivisions.

Chaque subdivision est elle-même subdivisée.

500 Sciences de la nature et Mathématiques ;

530 Physique ;

537 Électricité et électronique ;

537.2 Électrostatique ;

Inconvénients :

- information hiérarchisée suivant des critères parfois devenus obsolètes ;

Classification décimale de Dewey

Inventée au XIX^{e} siècle, elle range les ouvrages en 10 divisions.

Chaque division est rangée en 10 subdivisions.

Chaque subdivision est elle-même subdivisée.

500 Sciences de la nature et Mathématiques ;

530 Physique ;

537 Électricité et électronique ;

537.2 Électrostatique ;

Inconvénients :

- information hiérarchisée suivant des critères parfois devenus obsolètes ;
- exige du chercheur une connaissance précise de cette hiérarchisation.

Où trouver des ouvrages relatifs à la bio-informatique ?

à la côte 570 (Sciences de la vie, Biologie) ?
à la côte 620 (Art de l'ingénieur et activités connexes) ?
à la côte 000 (Généralités sur l'informatique) ?

Architecture d'une base de données

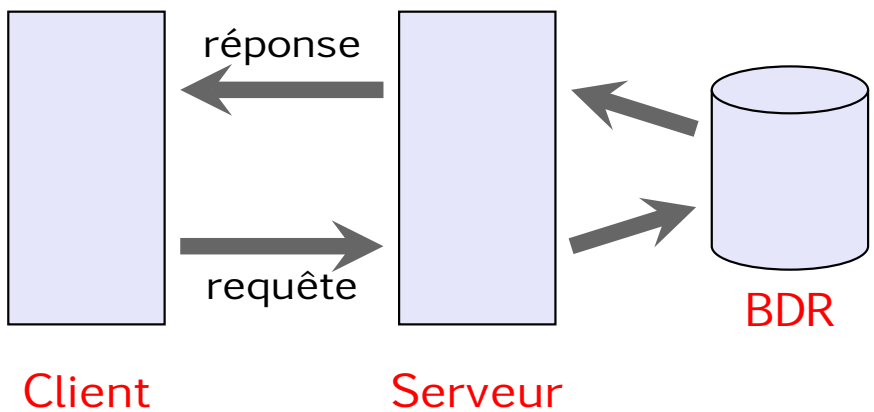
Les bases de données relationnelles organisent les données. Schématiquement, il s'agit d'un ensemble de tables contenant des données reliées entre elles par des relations ; on y extrait de l'information par le biais de requêtes exprimées dans un langage spécifique.

Architecture d'une base de données

Les bases de données relationnelles organisent les données. Schématiquement, il s'agit d'un ensemble de tables contenant des données reliées entre elles par des relations ; on y extrait de l'information par le biais de requêtes exprimées dans un langage spécifique.
Un système de gestion de bases de données est le logiciel qui organise et gère les données ; l'utilisateur interagit avec lui par l'intermédiaire de requêtes exprimées en SQL (*Structured Query Language*).

Architecture d'une base de données

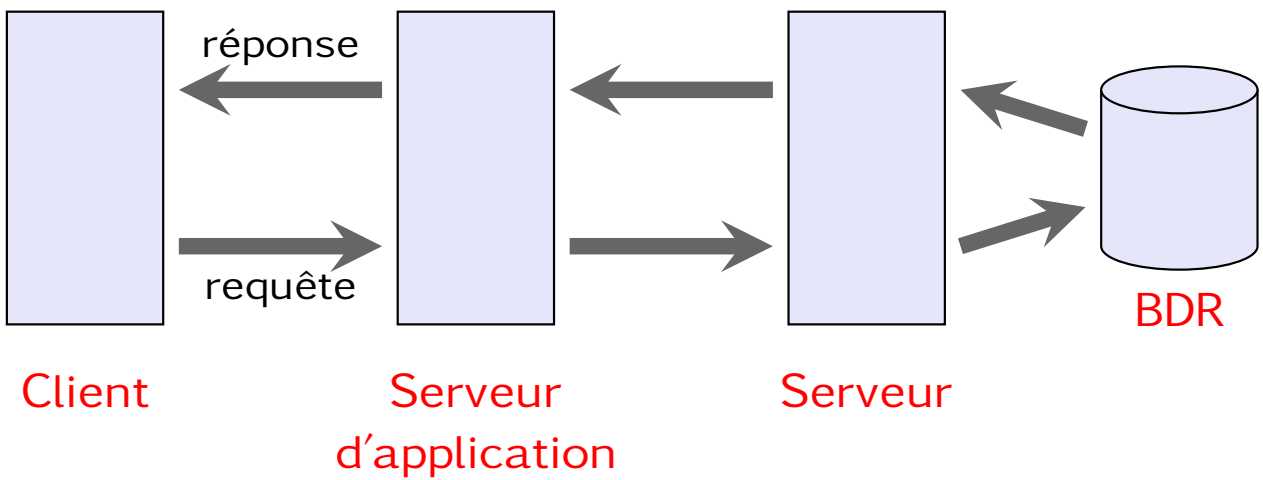
Les bases de données relationnelles organisent les données. Schématiquement, il s'agit d'un ensemble de tables contenant des données reliées entre elles par des relations ; on y extrait de l'information par le biais de requêtes exprimées dans un langage spécifique.
Un système de gestion de bases de données est le logiciel qui organise et gère les données ; l'utilisateur interagit avec lui par l'intermédiaire de requêtes exprimées en SQL (*Structured Query Language*).
Architecture deux-tiers :



Le logiciel client de la machine de l'utilisateur envoie des requêtes au logiciel serveur installé sur la machine qui traite les requêtes.

Architecture d'une base de données

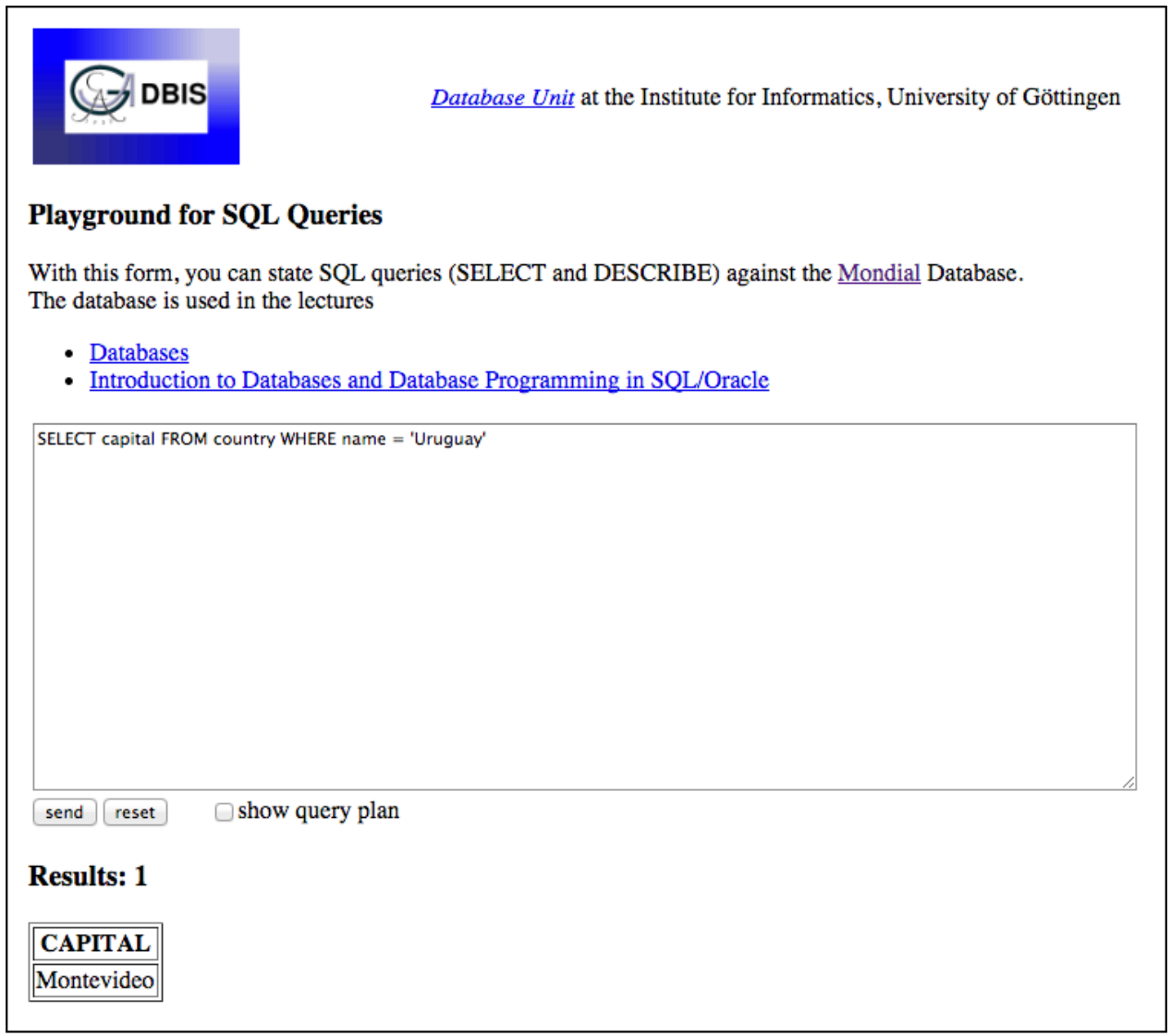
Les bases de données relationnelles organisent les données. Schématiquement, il s'agit d'un ensemble de tables contenant des données reliées entre elles par des relations ; on y extrait de l'information par le biais de requêtes exprimées dans un langage spécifique.
Un système de gestion de bases de données est le logiciel qui organise et gère les données ; l'utilisateur interagit avec lui par l'intermédiaire de requêtes exprimées en SQL (*Structured Query Language*).
Architecture trois-tiers :



L'architecture trois-tiers ajoute un niveau supplémentaire : le serveur de données transmet les informations à un serveur d'application qui, à son tour, transmet les informations traitées vers un client.

La base de données MONDIAL

- Accès par le biais d'un navigateur web.



La base de données MONDIAL

- Accès par le biais d'un navigateur web.
- ou accès direct aux données contenues dans un fichier.

```
Veuillez entrer une requête SQL (ou 'stop' pour terminer) :

SELECT capital FROM country WHERE name= 'Uruguay'
('Montevideo',)

Veuillez entrer une requête SQL (ou 'stop' pour terminer) :
```

La base de données MONDIAL

- Accès par le biais d'un navigateur web.
- ou accès direct aux données contenues dans un fichier.

```
Veuillez entrer une requête SQL (ou 'stop' pour terminer) :

SELECT capital FROM country WHERE name= 'Uruguay'
('Montevideo',)

Veuillez entrer une requête SQL (ou 'stop' pour terminer) :
```

Quelques différences :

- la base de donnée enregistrée localement est figée ; toute transformation ne peut qu'être locale.
- la base de donnée du web est interrogeable en SQL-Oracle ; la base de donnée locale en SQLite (différences mineures).

Relations

Une base de données est un ensemble de tables que l'on peut représenter sous la forme de tableaux bi-dimensionnels.

NAME	CODE	CAPITAL	PROVINCE	AREA	POPULATION
France	F	Paris	Ile de France	547030.	64933400
Spain	E	Madrid	Madrid	504750.	46815916
Austria	A	Wien	Wien	83850.	8499759
Czech Republic	CZ	Praha	Praha	78703.	10562214
Germany	D	Berlin	Berlin	356910.	80219695
Hungary	H	Budapest	Budapest	93030.	9937628
Italy	I	Roma	Lazio	301230.	59433744
Liechtenstein	FL	Vaduz	Liechtenstein	160.	36636

Les attributs désignent les éléments de chacune des colonnes ; les lignes en forment les enregistrements.

Relations

Une base de données est un ensemble de tables que l'on peut représenter sous la forme de tableaux bi-dimensionnels.

NAME	CODE	CAPITAL	PROVINCE	AREA	POPULATION
France	F	Paris	Ile de France	547030.	64933400
Spain	E	Madrid	Madrid	504750.	46815916
Austria	A	Wien	Wien	83850.	8499759
Czech Republic	CZ	Praha	Praha	78703.	10562214
Germany	D	Berlin	Berlin	356910.	80219695
Hungary	H	Budapest	Budapest	93030.	9937628
Italy	I	Roma	Lazio	301230.	59433744
Liechtenstein	FL	Vaduz	Liechtenstein	160.	36636

Les attributs désignent les éléments de chacune des colonnes ; les lignes en forment les enregistrements.
Pour croiser les données présentes dans plusieurs tables il faut posséder une caractérisation unique de chaque enregistrement d'une table ; c'est le rôle de la clé primaire (ici l'attribut Code).

Requêtes de base

Sélection de l'ensemble des valeurs d'un attribut :

```
SELECT name FROM country
```

Requêtes de base

Sélection de l'ensemble des valeurs d'un attribut :

```
SELECT name FROM country
```

Sélection de plusieurs attributs :

```
SELECT name, capital FROM country
```

Requêtes de base

Sélection de l'ensemble des valeurs d'un attribut :

```
SELECT name FROM country
```

Sélection de plusieurs attributs :

```
SELECT name, capital FROM country
```

Sélection de l'ensemble des attributs :

```
SELECT * FROM country
```

Requêtes de base

Sélection de l'ensemble des valeurs d'un attribut :

```
SELECT name FROM country
```

Sélection de plusieurs attributs :

```
SELECT name, capital FROM country
```

Sélection de l'ensemble des attributs :

```
SELECT * FROM country
```

Sélection de certains enregistrements uniquement :

```
SELECT capital FROM country WHERE name = 'Botswana'
```

Requêtes de base

Sélection de l'ensemble des valeurs d'un attribut :

```
SELECT name FROM country
```

Sélection de plusieurs attributs :

```
SELECT name, capital FROM country
```

Sélection de l'ensemble des attributs :

```
SELECT * FROM country
```

Sélection de certains enregistrements uniquement :

```
SELECT capital FROM country WHERE name = 'Botswana'
```

La valeur d'un attribut est **NULL** lorsque ce dernier est manquant :

```
SELECT name FROM country WHERE capital IS NULL
```

Requêtes de base

Résumé de la syntaxe SQL

SELECT *	sélection des colonnes
SELECT DISTINCT *	sélection sans doublon
FROM table	nom d'une table
WHERE condition	imposer une condition
GROUP BY expression	grouper les résultats
HAVING condition	condition sur un groupe
ORDER BY expression	trier les résultats
LIMIT n	limiter à *n* enregistrements (SQLite)
OFFSET n	débuter à partir de *n* enregistrements (SQLite)
OFFSET n **ROWS**	débuter à partir de *n* enregistrements (Oracle)
FETCH FIRST n **ROWS ONLY**	limiter à *n* enregistrements (Oracle)
UNION | **INTERSECT** | **EXCEPT**	opérations ensemblistes sur les requêtes

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;

```
SELECT name FROM country
WHERE population > 60000000
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;

```
SELECT name FROM country
WHERE population > 60000000
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;

```
SELECT name FROM country
WHERE population > 60000000
ORDER BY name
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;

```
SELECT name FROM country
WHERE population > 60000000
ORDER BY name
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;

```
SELECT name, population FROM country
WHERE population > 60000000
ORDER BY population DESC
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;
4. le nom des dix pays ayant la plus petite superficie ;

```
SELECT name, population FROM country
WHERE population > 60000000
ORDER BY population DESC
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;
4. le nom des dix pays ayant la plus petite superficie ;

```
SELECT name FROM country
ORDER BY area ASC FETCH FIRST 10 ROWS ONLY
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;
4. le nom des dix pays ayant la plus petite superficie ;
5. le nom des dix suivants.

```
SELECT name FROM country
ORDER BY area ASC FETCH FIRST 10 ROWS ONLY
```

Requêtes de base

Exercice

Rédiger une requête SQL pour obtenir :

1. la liste des pays dont la population excède 60 000 000 d'habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;
4. le nom des dix pays ayant la plus petite superficie ;
5. le nom des dix suivants.

```
SELECT name FROM country
ORDER BY area ASC OFFSET 10 ROWS FETCH FIRST 10 ROWS ONLY
```

Jointures

Réaliser une jointure permet de croiser des informations présentes dans plusieurs tables : si deux tables possèdent un attribut en commun, cet attribut peut servir de jointure entre elles.

Jointures

Réaliser une jointure permet de croiser des informations présentes dans plusieurs tables : si deux tables possèdent un attribut en commun, cet attribut peut servir de jointure entre elles.
Un extrait de la table country :

NAME	CODE	CAPITAL	PROVINCE	AREA	POPULATION
Bulgaria	BG	Sofia	Bulgaria	110910.	7284552
Romania	RO	Bucuresti	Bucuresti	237500.	20121641
Turkey	TR	Ankara	Ankara	780580.	75627384
Denmark	DK	Kobenhavn	Hovedstaden	43070.	5580516

Un extrait de la table encompasses :

COUNTRY	CONTINENT	PERCENTAGE
BG	Europe	100
RO	Europe	100
TR	Asia	97
TR	Europe	3
DK	Europe	100

Jointures

Réaliser une jointure permet de croiser des informations présentes dans plusieurs tables : si deux tables possèdent un attribut en commun, cet attribut peut servir de jointure entre elles.
On obtient la liste des pays européens par la requête :

```
SELECT c.name
FROM country c JOIN encompasses e
  ON c.code = e.country
WHERE e.continent = 'Europe'
```

Jointures

Réaliser une jointure permet de croiser des informations présentes dans plusieurs tables : si deux tables possèdent un attribut en commun, cet attribut peut servir de jointure entre elles.
On obtient la liste des pays européens par la requête :

```
SELECT c.name
FROM country c JOIN encompasses e
  ON c.code = e.country
WHERE e.continent = 'Europe'
```

Un extrait de la table :

country c **JOIN** encompasses e **ON** c.code = e.country

NAME	CODE	CAPITAL	PROVINCE	AREA	POPULATION	COUNTRY	CONTINENT	PERCENTAGE
Bulgaria	BG	Sofia	Bulgaria	110910.	7284552	BG	Europe	100
Romania	RO	Bucuresti	Bucuresti	237500.	20121641	RO	Europe	100
Turkey	TR	Ankara	Ankara	780580.	75627384	TR	Asia	97
Turkey	TR	Ankara	Ankara	780580.	75627384	TR	Europe	3
Denmark	DK	Kobenhavn	Hovedstaden	43070.	5580516	DK	Europe	100

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;

```sql
SELECT DISTINCT c.name
FROM country c JOIN encompasses e ON c.code = e.country
WHERE e.percentage < 100
```

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;
2. les pays du contient américain qui comptent moins de 10 habitants par km^2.

```
SELECT DISTINCT c.name
FROM country c JOIN encompasses e ON c.code = e.country
WHERE e.percentage < 100
```

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;
2. les pays du contient américain qui comptent moins de 10 habitants par km^2.

```sql
SELECT c.name
FROM country c JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'America'
  AND c.population / c.area < 10
```

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;
2. les pays du contient américain qui comptent moins de 10 habitants par km^2.
3. les capitales européennes situées à une latitude supérieure à 60°.

```sql
SELECT c.name
FROM country c JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'America'
  AND c.population / c.area < 10
```

Jointures

Exercice

Rédiger une requête SQL pour obtenir :

1. le nom des pays qui sont à cheval sur plusieurs continents ;
2. les pays du contient américain qui comptent moins de 10 habitants par km^2.
3. les capitales européennes situées à une latitude supérieure à 60°.

```
SELECT c.name, c.capital
FROM country c JOIN city v ON c.code = v.country
               JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'Europe'
  AND v.name = c.capital
  AND v.latitude > 60
```

Fonctions d'agrégation

On regroupe certains enregistrements d'une table par agrégation à l'aide du mot-clé **GROUP BY**.

COUNT( )	nombre d'enregistrements
MAX( )	valeur maximale d'un attribut
MIN( )	valeur minimale d'un attribut
SUM( )	calcul de la somme d'un attribut
AVG( )	calcul de la moyenne d'un attribut

Fonctions d'agrégation

On regroupe certains enregistrements d'une table par agrégation à l'aide du mot-clé **GROUP BY**.

COUNT( )	nombre d'enregistrements
MAX( )	valeur maximale d'un attribut
MIN( )	valeur minimale d'un attribut
SUM( )	calcul de la somme d'un attribut
AVG( )	calcul de la moyenne d'un attribut

Pour connaître le nombre de pays de chaque continent :

```
SELECT e.continent, COUNT(*)
FROM country c JOIN encompasses e ON c.code = e.country
GROUP BY e.continent
```

Fonctions d'agrégation

On regroupe certains enregistrements d'une table par agrégation à l'aide du mot-clé **GROUP BY**.

COUNT( )	nombre d'enregistrements
MAX( )	valeur maximale d'un attribut
MIN( )	valeur minimale d'un attribut
SUM( )	calcul de la somme d'un attribut
AVG( )	calcul de la moyenne d'un attribut

Pour connaître le nombre de pays de chaque continent :

```
SELECT e.continent, COUNT(*)
FROM country c JOIN encompasses e ON c.code = e.country
GROUP BY e.continent
```

HAVING impose des conditions sur un groupe. Pour obtenir la liste des continents dont la population totale dépasse le milliard d'habitants :

```
SELECT e.continent, SUM(c.population)
FROM country c JOIN encompasses e ON c.code = e.country
GROUP BY e.continent
  HAVING SUM(c.population) > 1000000000
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.

```
SELECT name, COUNT(*) c FROM language
GROUP BY name
ORDER BY c DESC
FETCH FIRST 10 ROWS ONLY
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ?

```
SELECT name, COUNT(*) c FROM language
GROUP BY name
ORDER BY c DESC
FETCH FIRST 10 ROWS ONLY
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ?

```
SELECT name FROM language
GROUP BY name
  HAVING COUNT(*) = 6
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?

```
SELECT name FROM language
GROUP BY name
  HAVING COUNT(*) = 6
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?

```
SELECT c.name AS pays, l.name AS langue
FROM country c JOIN language l ON c.code = l.country
WHERE l.name IN
  (SELECT name FROM language
   GROUP BY name
     HAVING COUNT(*) = 6)
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?
3. Quelles sont les langues parlées par moins de 30 000 personnes dans le monde ?

```
SELECT c.name AS pays, l.name AS langue
FROM country c JOIN language l ON c.code = l.country
WHERE l.name IN
  (SELECT name FROM language
   GROUP BY name
     HAVING COUNT(*) = 6)
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?
3. Quelles sont les langues parlées par moins de 30 000 personnes dans le monde ?

```
SELECT l.name
FROM language l JOIN country c ON l.country = c.code
GROUP BY l.name
  HAVING SUM(c.population * l.percentage / 100) < 30000
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?
3. Quelles sont les langues parlées par moins de 30 000 personnes dans le monde ?
4. Quelles sont les cinq langues les plus parlées en Afrique ? On précisera le nombre de personnes qui la parlent.

```
SELECT l.name
FROM language l JOIN country c ON l.country = c.code
GROUP BY l.name
  HAVING SUM(c.population * l.percentage / 100) < 30000
```

Fonctions d'agrégation

Exercice

1. Donner la liste ordonnée des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement six pays ? Et quels sont ces pays ?
3. Quelles sont les langues parlées par moins de 30 000 personnes dans le monde ?
4. Quelles sont les cinq langues les plus parlées en Afrique ? On précisera le nombre de personnes qui la parlent.

```
SELECT l.name, FLOOR(SUM(c.population * l.percentage / 100)) s
FROM language l JOIN country c ON l.country = c.code
                JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'Africa'
GROUP BY l.name
ORDER BY s DESC
FETCH FIRST 5 ROWS ONLY
```

Sous-requêtes

Il est possible d'imbriquer une requête dans une clause **SELECT**, ou (le plus souvent) au sein d'un filtre **WHERE** ou **HAVING**.
Pour déterminer les pays dont la densité de population est supérieure à la moyenne :

```
SELECT name FROM country
WHERE population / area >
      (SELECT AVG(population / area) FROM country)
```

Sous-requêtes

Exercice

1. Déterminer les pays majoritairement agricoles dont le taux de chômage est inférieur à la moyenne mondiale.

Sous-requêtes

Exercice

1. Déterminer les pays majoritairement agricoles dont le taux de chômage est inférieur à la moyenne mondiale.

```
SELECT c.name
FROM country c JOIN economy e ON c.code = e.country
WHERE e.agriculture > e.service
  AND e.agriculture > e.industry
  AND e.unemployment < (SELECT AVG(unemployment) FROM economy)
```

Sous-requêtes

Exercice

1. Déterminer les pays majoritairement agricoles dont le taux de chômage est inférieur à la moyenne mondiale.
2. Déterminer pour chaque continent le pays au taux d'inflation le plus faible parmi les pays majoritairement industriels.

```
SELECT c.name
FROM country c JOIN economy e ON c.code = e.country
WHERE e.agriculture > e.service
  AND e.agriculture > e.industry
  AND e.unemployment < (SELECT AVG(unemployment) FROM economy)
```

Sous-requêtes

Exercice

1. Déterminer les pays majoritairement agricoles dont le taux de chômage est inférieur à la moyenne mondiale.
2. Déterminer pour chaque continent le pays au taux d'inflation le plus faible parmi les pays majoritairement industriels.

```
SELECT en.continent, c.name
FROM country c JOIN economy e ON c.code = e.country
               JOIN encompasses en ON en.country = c.code
WHERE e.industry > e.agriculture
  AND e.industry > e.service
  AND (en.continent, e.inflation) IN
      (SELECT en.continent, MIN(e.inflation)
       FROM economy e JOIN encompasses en ON e.country = en.country
       WHERE e.industry > e.agriculture AND e.industry > e.service
       GROUP BY en.continent)
```

Algèbre relationnelle

Opérations ensemblistes

Trois opérations ensemblistes peuvent être effectuées avec les relations : les opérations d'union ($\cup$), d'intersection ($\cap$) et de différence ($-$).

R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2

A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 \cup R_2$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

$R_1 \cap R_2$

A	B	C
a_1	b_1	c_1

$R_1 - R_2$

A	B	C
a_2	b_2	c_2

Pour être réalisées, ces opérations doivent agir sur des relations ayant le même schéma relationnel.

Algèbre relationnelle

Projection, sélection

La projection extrait une relation d'une relation donnée en supprimant des attributs de cette dernière.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_1	b_1	c_3

$\pi_{(A,B)}(R)$

A	B
a_1	b_1
a_2	b_2

Algèbre relationnelle

Projection, sélection

La projection extrait une relation d'une relation donnée en supprimant des attributs de cette dernière.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_1	b_1	c_3

$\pi_{(A,B)}(R)$

A	B
a_1	b_1
a_2	b_2

La sélection permet d'extraire les enregistrements d'une relation R qui satisfont une expression logique E.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4

$\sigma_E(R)$

A	B	C
a_2	b_2	c_2
a_4	b_4	c_4

Algèbre relationnelle

Renommage, jointure

Le renommage permet la modification du nom d'un ou plusieurs attributs d'une relation.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

$\rho_{a \leftarrow d}(R)$

D	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

Algèbre relationnelle

Renommage, jointure

Le renommage permet la modification du nom d'un ou plusieurs attributs d'une relation.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

$\rho_{a \leftarrow d}(R)$

D	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

La jointure porte sur deux relations R_1 et R_2 et retourne une relation qui comporte les enregistrements des deux premières relations qui satisfont une contrainte logique E.

R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R_2

D	E
a_1	e_1
a_2	e_2

$R_1 \bowtie_{A=D} R_2$

A	B	C	E
a_1	b_1	c_1	e_1
a_2	b_2	c_2	e_2

Algèbre relationnelle

Produit et division cartésiens

Il est possible de réaliser 1e produit cartésien de deux relations R_1 et R_2 :

R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2

D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

Algèbre relationnelle

Produit et division cartésiens

Il est possible de réaliser 1e produit cartésien de deux relations R_1 et R_2 :

R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2

D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

La division cartésienne produit une relation à partir de deux relations R_1 et R_2 vérifiant $R_2 \subset R_1$. La relation obtenue possède tous les attributs de R_1 que ne possède pas R_2.

R_1

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_3	d_3
a_3	b_3	c_1	d_1
a_3	b_3	c_2	d_2

R_2

C	D
c_1	d_1
c_2	d_2

$R_1 \div R_2$

A	B
a_1	b_1
a_3	b_3

Algèbre relationnelle

Exercices

Donner un sens aux expressions de l'algèbre relationnelle suivantes :

1. $\pi_{Name}(\sigma_{Latitude>66}(\text{city}) \cap \sigma_{Population>10000}(\text{city}))$;

Algèbre relationnelle

Exercices

Donner un sens aux expressions de l'algèbre relationnelle suivantes :

1. $\pi_{Name}(\sigma_{Latitude>66}(\text{city}) \cap \sigma_{Population>10000}(\text{city}))$;

Villes de plus de 10 000 habitants situées au delà du cercle polaire arctique.

Algèbre relationnelle

Exercices

Donner un sens aux expressions de l'algèbre relationnelle suivantes :

1. $\pi_{Name}(\sigma_{Latitude>66}(\text{city}) \cap \sigma_{Population>10000}(\text{city}))$;
2. $\pi_{country.Name}(\sigma_{city.Latitude<0}(x) - \sigma_{city.Latitude>-23}(x))$ avec $x = \text{country} \bowtie_{country.code=city.country} \text{city}$.

Villes de plus de 10 000 habitants situées au delà du cercle polaire arctique.

Algèbre relationnelle

Exercices

Donner un sens aux expressions de l'algèbre relationnelle suivantes :

1. $\pi_{Name}(\sigma_{Latitude>66}(\text{city}) \cap \sigma_{Population>10000}(\text{city}))$;
2. $\pi_{country.Name}(\sigma_{city.Latitude<0}(x) - \sigma_{city.Latitude>-23}(x))$ avec $x = \text{country} \bowtie_{country.code=city.country} \text{city}$.

Villes de plus de 10 000 habitants situées au delà du cercle polaire arctique.

Nom des pays qui possèdent au moins une ville située entre l'équateur et le tropique du Capricorne.