BASES DE DONNÉES

B. Landelle

Table des matières

I Introduction

1 Généralités

Pour gérer de grands volumes de données, on utilise des *bases de données* (*Database* en anglais) qui permettent d'organiser les données, de les modifier et d'en extraire des informations ciblées de manière optimisée.

Par exemple, pour une bibliothèque, il serait contre-productif, de créer, lors de chaque emprunt, un dossier contenant toutes les informations de l'emprunteur, toutes les informations du livre et la date d'emprunt. Il est beaucoup plus pertinent d'avoir une table contenant les informations de tous les adhérents, une table contenant les informations de tous les livres et une table d'emprunt dans laquelle on enregistre les référence de l'emprunteur et du livre emprunté pour une date donnée.

Définition 1. *Une* base de donnée *est une structure conçue pour le stockage, la consultation et la modification d'informations.*

2 Architecture

Les bases de données sont utilisées dans des architectures distribuées.

Définition 2. *Une architecture* client-serveur *désigne un environnement où le* client *et le* serveur *sont des processus ou des programmes qui effectuent des transactions selon le schéma suivant : le client soumet des requêtes au serveur et le serveur répond aux requêtes.*

Concrètement, une architecture client-serveur peut exister sur une même machine où des processus ou programmes sont respectivement clients et serveurs. Il est aussi fréquent que l'architecture soit distribuée sur plusieurs postes : un serveur sur un ordinateur avec d'importantes ressources de calculs et stockages et des machines clientes qui interrogent le serveur.

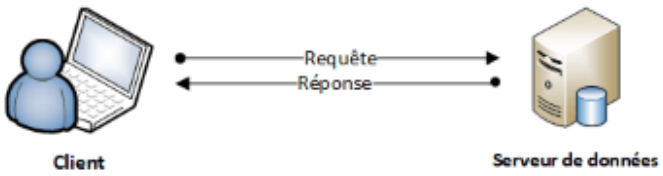


FIGURE 1 – Architecture client-serveur

Définition 3. *Une architecture* trois-tiers *désigne un environnement où le* client *communique avec un* serveur *de données via un serveur* applicatif.

Cette architecture permet de spécialiser les rôles et offre donc plus de sécurité et plus de flexibilité. Le serveur de données est dédié à un certain types d'applications, par exemple un système de bases de données et c'est le serveur applicatif qui assure la liaison entre clients et serveur.

FIGURE 2 – Architecture trois-tiers

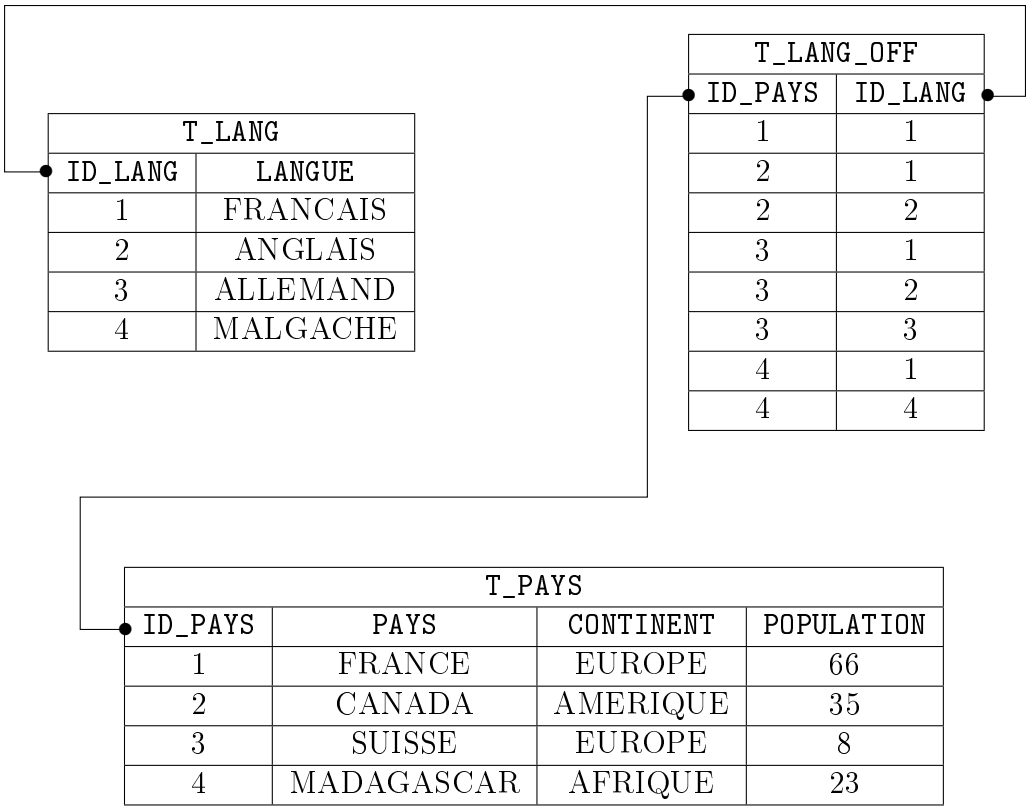
3 Système de Gestion de Bases de Données

Définition 4. *Un* système de gestion de bases de données *(SGBD) est un logiciel qui permet de stocker et gérer des bases de données.*

Dans le cadre de ce cours, nous utiliserons `DB Browser`.

II Organisation

Dans ce qui suit, on illustrera les différentes notions présentées sur l'exemple très simple de la base de données constituée des trois tables suivantes :



T_LANG

ID_LANG	LANGUE
1	FRANCAIS
2	ANGLAIS
3	ALLEMAND
4	MALGACHE

T_LANG_OFF

ID_PAYS	ID_LANG
1	1
2	1
2	2
3	1
3	2
3	3
4	1
4	4

T_PAYS

ID_PAYS	PAYS	CONTINENT	POPULATION
1	FRANCE	EUROPE	66
2	CANADA	AMERIQUE	35
3	SUISSE	EUROPE	8
4	MADAGASCAR	AFRIQUE	23

La table `T_LANG` contient une liste de langues officielles, la table `T_PAYS` contient une liste de pays et la table `T_LANG_OFF` contient la correspondance entre langues officielles et pays référencés dans les autres tables.

Pour la table `T_LANG_OFF`, la première ligne signifie que en France `(ID_PAYS=1)`, le français `(ID_LANG=1)` est langue officielle. Les autres lignes suivent le même principe.

1 Relations, attributs

Les *bases de données relationnelles* sont organisées en *tables* à deux dimensions aussi appelées *relations*.

Définition 5. *Le SQL (Structured Query Langage) est un langage informatique normalisé servant à exploiter des bases de données relationnelles.*

Définition 6. *Une* relation *est un ensemble de n-uplets.*

Exemple : La relation `T_PAYS` est constituée des 4-uplets (1,FRANCE,EUROPE,66), (2,CANADA,AMERIQUE,35), etc.

Une relation correspond à une table dans une base de données et les n-uplets sont les enregistrements ou lignes de la table.

Définition 7. *Les colonnes d'une relation sont ses* attributs. *Les attributs peuvent être au format :*
- *entier (INT, BIGINT, ...) ;*
- *numérique (FLOAT, DECIMAL, ...) ;*
- *texte (TEXT, CHAR, ...) ;*
- *date (DATE, YEAR, ...) ;*
- *etc. ...*

Exemple : Les attributs de `T_PAYS` sont `ID_PAYS`, `PAYS`, `CONTINENT`, `POPULATION`.

2 Clés primaires, clés étrangères

Définition 8. *Une* clé primaire *est une donnée qui permet d'identifier de manière unique un n-uplet d'une relation. Une clé primaire est un attribut ou un ensemble d'attribut de la relation considérée.*

Exemple : L'attribut `ID_LANG` est une clé primaire de la table `T_LANG`, l'attribut `ID_PAYS` est une clé primaire de la table `T_PAYS` et le couple d'attributs `(ID_PAYS, ID_LANG)` est une clé primaire de la table `T_LANG_OFF`.

Définition 9. *Une* clé étrangère *est une donnée qui garantit l'intégrité référentielle entre deux tables. Une clé étrangère est un attribut ou un ensemble d'attribut de la relation considérée.*

Exemples : 1. Si on reprend l'exemple de la base de données d'une bibliothèque, on ne doit pas pouvoir créer d'emprunt pour un livre ou un emprunteur qui ne serait pas référencé. Afin de garantir cette intégrité, on définit ces attributs comme clés étrangères dans la table d'emprunt.

2. Dans la table `T_LANG_OFF`, on définit les attributs `ID_PAYS` et `ID_LANG` comme clés étrangères : pour définir les langues officielles par pays, il faut que les langues et les pays en question soient référencés.

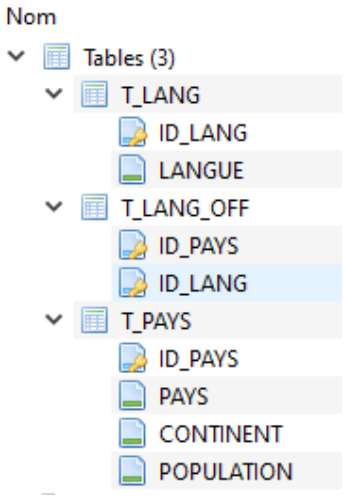


FIGURE 3 – Structure des tables

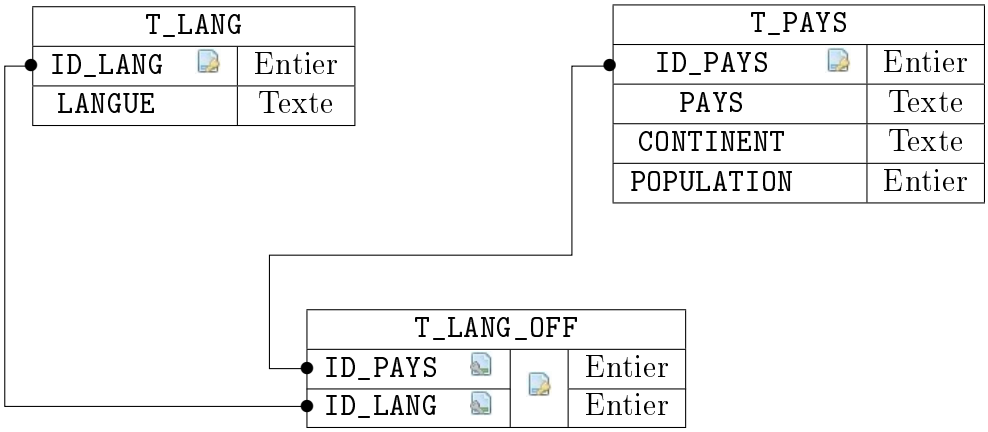


FIGURE 4 – Visualisation des clés primaires et étrangères

3 Entités et associations

Définition 10. *Dans une base de données, les* entités *sont les éléments organisés en table et les* associations *sont des relations entre les entités. On distingue les associations de type :*

- `1-1` *: une entité associée à une et une seule entité ;*
- `1-*` *: une entité associée à aucune ou une ou plusieurs entités ;*
- `*-*` *: aucune ou une ou plusieurs entités associées à aucune ou une ou plusieurs entités.*

Exemple : Dans la base proposée en exemple, les entités sont la langue, la langue officielle et le pays. Une langue est langue officielle de un ou plusieurs pays et un pays possède une ou plusieurs langues officielles. Les associations entre `T_LANG_OFF` et `T_LANG` et entre `T_LANG_OFF` et `T_PAYS` sont de type `1-*` : une langue officielle (à savoir un couple (langue,pays)) correspond à une langue et un pays, une langue peut être plusieurs fois langue officielle et un pays peut avoir plusieurs langues officielles.

Proposition 1. *On a les traductions des associations suivantes :*

- `1-*` *: utilisation d'une clé étrangère d'une table en lien avec une clé primaire d'une autre ;*
- `1-1` *: utilisation d'une clé étrangère d'une table en lien avec une clé primaire d'une autre et réciproquement entre les tables ;*
- `*-*` *: l'association est séparée en deux associations* `1-*` *avec utilisation d'une table faisant le lien.*

Exemples : 1. Dans la base proposée en exemple, l'association `1-*` entre `T_LANG_OFF` et `T_LANG` se traduit par une clé primaire `ID_LANG` dans `T_LANG` et une clé étrangère `ID_LANG` dans `T_LANG_OFF`.

2. On considère le modèle simpliste :

Auteurs	
id_auteur	Entier
id_livre	Entier
nom	Texte
prenom	Texte

Livres	
id_livre	Entier
id_auteur	Entier
titre	Texte

où un auteur écrit un livre et un seul. On choisit pour la table `Auteurs` :

— `id_auteur` comme clé primaire ;
— `id_livre` comme clé étrangère ;

et pour la table `Livres` :

— `id_livre` comme clé primaire ;
— `id_auteur` comme clé étrangère.

3. On considère une base utilisée pour enregistrer des écrivains et des livres avec la possibilité d'avoir un livre ayant plusieurs co-auteurs. On a une association `*-*` entre écrivains et livres : un écrivain peut être auteur de plusieurs livres et un livre peut avoir plusieurs co-auteurs. L'association est séparée en deux associations `1-*` avec par exemple les tables :

Ecrivains	
id_ecrivain	Entier

Auteurs	
id_ecrivain	Entier
id_livre	Entier

Livres	
id_livre	Entier

avec les attributs de `Auteurs` clés étrangères en relation avec les clés primaires que sont les attributs éponymes des tables `Ecrivains` et `Livres`.

Exercice : On considère la base de donnée constituée des tables suivantes :

Emprunteurs	
id_emprunteur	Entier
nom	Texte
prenom	Texte

Livres	
id_livre	Entier
id_auteur	Entier
titre	Texte

Auteurs	
id_auteur	Entier
nom	Texte
prenom	Texte

Emprunts	
id_livre	Entier
id_emprunteur	Entier
date_emprunt	Date
date_retour	Date

Un livre admet donc un seul auteur. Préciser les associations entre entités puis identifier des attributs pertinents pour être :

— des clés primaires ;
— des clés étrangères.

Justifier vos choix.

Corrigé : Un emprunteur peut emprunter plusieurs livres. Un livre peut être emprunté plusieurs fois. Un emprunt correspond à un livre et un auteur (et une date). Enfin, un auteur peut écrire plusieurs livres et un livre admet un seul auteur. On en déduit les associations :

— `1-*` entre emprunteurs et emprunts ;
— `1-*` entre livres et emprunts ;
— `1-*` entre auteur et livre.

Pour chaque table, on choisit une clé primaire permettant d'identifier de manière unique un enregistrement :

— `id_emprunteur` pour `Emprunteurs` ;
— `id_livre` pour `Livres` ;
— `id_auteurs` pour `Auteurs` ;
— le triplet `(id_livre, id_emprunteur, date_emprunt)` pour `Emprunts`.

On choisit également des clés étrangères pour garantir l'intégrité référentielle. Ainsi, un livre est écrit par un auteur donc l'auteur doit être référencé dans la table Auteurs. De même, un emprunt concerne un emprunteur et un livre et il faut donc que chacun soit référencé dans la table correspondante. On en déduit les choix suivants de clés étrangères :

— `id_auteur` pour Livres ;
— `id_livre` pour Emprunts ;
— `id_emprunteur` pour Emprunts.

III Requêtes simples

1 Projection

Proposition 2. *L'instruction* `SELECT attribut FROM table` *renvoie la projection d'un attribut dans une table.*

Remarque : Avec `*`, on sélectionne tous les attributs.

Requête : On affiche le contenu de la table `T_PAYS`.

```
SELECT * FROM T_PAYS
```

Résultat :

ID_PAYS	PAYS	CONTINENT	POPULATION
1	FRANCE	EUROPE	66
2	CANADA	AMERIQUE	35
3	SUISSE	EUROPE	8
4	MADAGASCAR	AFRIQUE	23

Requête : On affiche les champs `PAYS` et `CONTINENT` de la table `T_PAYS`.

```
SELECT PAYS,CONTINENT
FROM T_PAYS
```

Résultat :

PAYS	CONTINENT
FRANCE	EUROPE
CANADA	AMERIQUE
SUISSE	EUROPE
MADAGASCAR	AFRIQUE

Proposition 3. *L'instruction* `ORDER BY` *réalise le tri d'une sélection selon un attribut donné. L'option* `DESC` *permet un tri décroissant.*

Requête : On affiche les champs `PAYS` et `POPULATION` de la table `T_PAYS` et on trie par population croissante.

```
SELECT PAYS,POPULATION
FROM T_PAYS
ORDER BY POPULATION
```

Résultat :

PAYS	POPULATION
SUISSE	8
MADAGASCAR	23
CANADA	35
FRANCE	66

Proposition 4. *L'instruction* `LIMIT` *limite la taille d'une sélection et l'instruction* `OFFSET` *décale une sélection limitée.*

Requête : On affiche les deux premiers pays les plus peuplés de la table `PAYS`.

```
-- AFFICHAGE DES DEUX PREMIERS PAYS LES PLUS PEUPLES
SELECT PAYS FROM T_PAYS
ORDER BY POPULATION DESC LIMIT 2
```

Résultat :

PAYS
FRANCE
CANADA

On affiche le pays le plus peuplé après les deux premiers les plus peuplés de la table `PAYS`.

```
-- AFFICHAGE DU PAYS APRES LES DEUX LES PLUS PEUPLES
SELECT PAYS FROM T_PAYS
ORDER BY POPULATION DESC LIMIT 1 OFFSET 2
```

Résultat :

PAYS
MAGADASCAR

Requête : On affiche le champ `CONTINENT` de la table `T_PAYS`. On constate qu'il y a des doublons.

```
-- AFFICHAGE DES CONTINENTS
SELECT CONTINENT
FROM T_PAYS
```

Résultat :

CONTINENT
EUROPE
AMERIQUE
EUROPE
AFRIQUE

Proposition 5. *L'instruction* `DISTINCT` *élimine les doublons d'une sélection.*

Requête : On affiche le champ `CONTINENT` sans doublon de la table `T_PAYS` et on les trie par ordre alphabétique.

```
-- AFFICHAGE DES CONTINENTS SANS DOUBLON DANS L'ORDRE ALPHABETIQUE
SELECT DISTINCT CONTINENT
FROM T_PAYS
ORDER BY CONTINENT
```

Résultat :

CONTINENT
AFRIQUE
AMERIQUE
EUROPE

On verra une variante avec des fonctions d'agrégation.

Proposition 6. *On dispose des opérations ensemblistes suivantes :*
- `UNION` *pour l'union ;*
- `INTERSECT` *pour l'intersection ;*
- `EXCEPT` *pour privé de ;*
- `CROSS JOIN` *ou simplement* `,` *pour le produit cartésien.*

Exercice : Afficher le contenu de la table `T_LANG` puis afficher uniquement les langues de cette table.

Corrigé : On saisit :

```
-- CONTENU DE T_LANG
SELECT * FROM T_LANG;

-- LANGUES DANS T_LANG
SELECT LANGUE FROM T_LANG
```

2 Sélection

Proposition 7. *L'instruction* `WHERE condition` *effectue la sélection des lignes d'une table vérifiant la condition souhaitée.*

Proposition 8. *Dans une sélection, on utilise les opérateurs :*
- `=` *: test d'égalité ;*
- `<>` *: test de différence ;*
- `<, <=, >, >=` *: test de relation d'ordre ;*
- `IN` *: test d'appartenance ;*
- `LIKE` *: test de format ;*
- `IS` *: test d'égalité.*

Remarque : Tester un attribut vide s'écrit `IS NULL` et non avec l'opérateur `=`.

Requête : On affiche le champ `PAYS` de la table `T_PAYS` dont le continent est EUROPE.

```
-- PAYS D'EUROPE
SELECT PAYS
FROM T_PAYS
WHERE CONTINENT='EUROPE'
```

Résultat :

PAYS
FRANCE
SUISSE

Requête : On affiche les langues officielles commençant par la lettre `"A"`.

```
-- LANGUES COMMENCANT PAR A
SELECT LANGUE
FROM T_LANG
WHERE LANGUE LIKE "A%"
```

Résultat :

LANGUE
ANGLAIS
ALLEMAND

On peut moduler l'usage du `%`. Par exemple, pour afficher les langues contenant la lettre `"L"`, on remplace la sélection précédente par `WHERE LANGUE LIKE "%L%"`.

Requête : On affiche les pays des continents européen et africain. Plusieurs approches sont possibles :

```
-- AFFICHAGE DES PAYS DES CONTINENTS EUROPEEN ET AFRICAIN
SELECT PAYS FROM T_PAYS
WHERE CONTINENT IN("EUROPE","AFRIQUE")
```

ou

```
SELECT PAYS FROM T_PAYS
WHERE CONTINENT IS "EUROPE" OR CONTINENT IS "AFRIQUE"
```

ou avec une union :

```
SELECT PAYS FROM T_PAYS
WHERE CONTINENT IS "EUROPE"
UNION
SELECT PAYS FROM T_PAYS
WHERE CONTINENT IS "AFRIQUE"
```

Résultat :

PAYS
FRANCE
MAGADASCAR

Requête : On affiche les pays dont la population dépasse 10 millions d'habitants et qui ne sont pas sur le continent américain. Plusieurs approches sont possibles :

```
-- PAYS HORS AMERIQUE DE PLUS DE 10 MILLIONS D'HABITANTS
SELECT PAYS FROM T_PAYS
WHERE CONTINENT<>"AMERIQUE" AND POPULATION>10
```

ou

```
SELECT PAYS FROM T_PAYS WHERE POPULATION>10
EXCEPT
SELECT PAYS FROM T_PAYS WHERE CONTINENT="AMERIQUE"
```

Exercice : Afficher les pays d'Europe dont la population dépasse 30 millions d'habitants.

Corrigé : On saisit :

```
-- PAYS D'EUROPE DE PLUS DE 30 MILLIONS D'HABITANTS
SELECT PAYS FROM T_PAYS
WHERE CONTINENT="EUROPE" AND POPULATION>30
```

3 Fonctions d'agrégation

Définition 11. *Une* fonction d'agrégation *est une fonction d'un ou plusieurs attributs d'une relation.*

Proposition 9. *L'instruction* `GROUP BY attribut` *regroupe les enregistrements par* `attribut`. *L'instruction* `HAVING condition` *permet de faire une sélection après un regroupement.*

Remarque : Sous `DB Browser`, le `HAVING` ne peut être utilisé que consécutivement à un `GROUP BY`. Sous d'autres SGBD, il est possible d'utiliser le `HAVING` sans appel préalable à un `GROUP BY` : celui-ci se comporte alors comme un `WHERE`.

Proposition 10. *Les fonctions* `MAX, MIN, COUNT, SUM, AVG` *sont les fonctions d'agrégations correspondant aux fonctions éponymes (*`AVG` *pour* average*).*

Requête : On affiche le pays dont la population est maximale.

```
-- PAYS DE POPULATION MAXIMALE
SELECT PAYS, MAX(POPULATION) AS POP_MAX
FROM T_PAYS
```

Résultat :

PAYS	POP_MAX
FRANCE	66

Requête : On affiche la population totale des pays d'Europe.

```
-- POPULATION TOTALE EN EUROPE
SELECT SUM(POPULATION) AS POP_EUROPE
FROM T_PAYS
WHERE CONTINENT='EUROPE'
```

Résultat :

POP_EUROPE
74

Requête : On affiche le champ `CONTINENT` sans doublon de la table `T_PAYS` et on les trie par ordre alphabétique.

```
-- AFFICHAGE DES CONTINENTS SANS DOUBLON DANS L'ORDRE ALPHABETIQUE
SELECT CONTINENT
FROM T_PAYS
GROUP BY CONTINENT
ORDER BY CONTINENT
```

Résultat :

CONTINENT
AFRIQUE
AMERIQUE
EUROPE

Requête : On affiche le nombre de continents de la table `T_PAYS`.

```
-- NOMBRE DE CONTINENTS
SELECT COUNT(DISTINCT CONTINENT) AS NB_CONT
FROM T_PAYS
```

Résultat :

NB_CONT
3

On verra une variante avec sous-requête.

Requête : On affiche le champ `CONTINENT` de la table `T_PAYS` en rassemblant les doublons et en triant par nombre d'occurences décroissant de ces doublons (autrement dit du continent le plus représenté au continent le moins représenté).

```
-- CONTINENT DU PLUS REPRESENTE AU MOINS REPRESENTE
SELECT CONTINENT, COUNT(*) AS NB_PAYS
FROM T_PAYS
GROUP BY CONTINENT
ORDER BY NB_PAYS DESC
```

Résultat :

CONTINENT	NB_PAYS
EUROPE	2
AFRIQUE	1
AMERIQUE	1

Exercice : Écrire une requête déterminant les continents dont la population dépasse 30 millions d'habitants. On affichera les continents concernés et leurs populations.

Corrigé :

```
-- CONTINENT DE PLUS DE 30 MILLIONS D'HAB
SELECT CONTINENT, SUM(POPULATION) AS POP
FROM T_PAYS
GROUP BY CONTINENT
HAVING POP>30
```

IV Requêtes élaborées

1 Requêtes avec sous-requêtes

Proposition 11. *Une requête peut utiliser une ou plusieurs sous-requêtes.*

Requête : On affiche pays et populations quand celles-ci dépassent la population moyenne des pays. On effectue une sous-requête dans la requête principale pour déterminer la population moyenne des pays enregistrés.

```
-- PAYS DONT POPULATION DEPASSE LA MOYENNE
SELECT PAYS, POPULATION
FROM T_PAYS
WHERE POPULATION>(SELECT AVG(POPULATION) FROM T_PAYS)
```

Résultat :

PAYS	POPULATION
FRANCE	66
CANADA	35

Requête : On affiche le nombre de continents de la table `T_PAYS`.

```
SELECT COUNT(*) AS NB_CONT
FROM (SELECT * FROM T_PAYS GROUP BY CONTINENT)
```

Résultat :

NB_CONT
3

Exercice : Écrire une requête déterminant les pays du monde dont la population est inférieure à la population moyenne en Europe. On affichera pays et populations.

```
-- PAYS DE POPULATION< POP MOYENNE D'EUROPE
SELECT PAYS, POPULATION
FROM T_PAYS
WHERE POPULATION<
(SELECT AVG(POPULATION) FROM T_PAYS WHERE CONTINENT='EUROPE')
```

2 Requêtes avec jointures

Proposition 12. *L'instruction* `table1 JOIN table2 ON id1=id2` *réalise une jointure de* `table1` *sur* `table2` *où la liaison se fait par les attributs* `id1` *et* `id2`.

Requête : Avec une jointure, on affiche les identifiants des pays avec leurs langues officielles.

```
-- ID_PAYS ET LANGUE OFFICIELLE
    SELECT ID_PAYS, LANGUE
    FROM T_LANG JOIN T_LANG_OFF
    ON T_LANG.ID_LANG=T_LANG_OFF.ID_LANG
```

Résultat :

ID_PAYS	LANGUE
1	FRANCAIS
2	FRANCAIS
2	ANGLAIS
3	FRANCAIS
3	ANGLAIS
3	ALLEMAND
4	FRANCAIS
4	MALGACHE

On peut écrire cette requête en exploitant le renommage ce qui rend sa syntaxe plus légère.

```
-- ID_PAYS ET LANGUE OFFICIELLE
    SELECT ID_PAYS, LANGUE
    FROM T_LANG AS L JOIN T_LANG_OFF AS LO
    ON L.ID_LANG=LO.ID_LANG
```

Remarque : On pourrait aussi écrire une sélection :

```
SELECT ID_PAYS, LANGUE
FROM T_LANG AS L, T_LANG_OFF AS LO
WHERE L.ID_LANG=LO.ID_LANG
```

Quand les deux situations sont possibles, à savoir jointure ou sélection, on privilégie la jointure dont l'exécution est optimisée.

Requête : Avec une jointure, on affiche les pays admettant exactement trois langues officielles.

```
-- PAYS AVEC TROIS LANGUES OFFICIELLES
SELECT PAYS
FROM T_PAYS AS P
JOIN T_LANG_OFF AS LO
ON P.ID_PAYS=LO.ID_PAYS
GROUP BY PAYS
HAVING COUNT(*)=3
```

Résultat :

PAYS
SUISSE

Exercice : Écrire une requête déterminant les langues officielles parlées sur au moins deux pays.

```
-- LANGUES PARLEES SUR PLUS QU'UN PAYS
SELECT LANGUE, COUNT(*) AS NB_PAYS
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
GROUP BY LO.ID_LANG
HAVING NB_PAYS>1
```

Exercice : Écrire une requête déterminant pour chaque langue, le nombre de pays où celle-ci est officielle. On affichera langue et nombre de pays, de la plus répandue à la moins répandue.

Corrigé : On saisit :

```
-- LANGUES DE LA PLUS REPANDUE A LA MOINS REPANDUE
SELECT LANGUE, COUNT(*) AS NB_PAYS
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
GROUP BY LANGUE
ORDER BY NB_PAYS DESC
```

Exercice : Écrire une requête déterminant la langue dont le nombre de pays où celle-ci est officielle est maximal. On affichera la langue concernée et le nombre de pays associés.

Corrigé : On saisit :

```
-- LANGUES LA PLUS REPANDUE PAR PAYS
SELECT LANGUE, COUNT(*) AS NB_PAYS
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
GROUP BY LANGUE
HAVING NB_PAYS=
(SELECT MAX(NBP) FROM
(SELECT COUNT(*) AS NBP
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
GROUP BY LANGUE))
```

On peut procéder plus simplement en affichant chaque langue avec le nombre de pays l'ayant pour langue officielle, par ordre décroissant de de ce nombre de pays et en limitant l'affichage au premier résultat avec l'instruction `LIMIT`.

```
-- LANGUES LA PLUS REPANDUE PAR PAYS
SELECT LANGUE, COUNT(*) AS NB_PAYS
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
GROUP BY LANGUE
ORDER BY NB_PAYS DESC LIMIT 1
```

Proposition 13. *L'instruction* `table1 JOIN table2 ON id1=id2 JOIN table3 on id2=id3 JOIN ...` *réalise des jointures multiples de* `table1` *sur* `table2` *où la liaison se fait par les attributs* `id1` *et* `id2` *puis de* `table2` *sur* `table3` *où la liaison se fait par les attributs* `id2` *et* `id3` *etc. ...*
On peut aussi écrire toutes les jointures puis toutes les liaisons avec des conjonctions entre elles : `table1 JOIN table2 JOIN table3 JOIN ... ON id1=id2 AND id2=id3 AND ...`

Requête : Avec des jointures, on affiche les pays dont la langue officielle est l'anglais.

```
-- PAYS DONT LA LANGUE OFFICIELLE EST L'ANGLAIS
SELECT PAYS
FROM T_PAYS AS P
JOIN T_LANG_OFF AS LO
ON P.ID_PAYS=LO.ID_PAYS
JOIN T_LANG AS L
ON LO.ID_LANG=L.ID_LANG
WHERE L.LANGUE="ANGLAIS"
```

ou aussi

```
SELECT PAYS
FROM T_PAYS AS P JOIN T_LANG_OFF AS LO JOIN T_LANG AS L
ON P.ID_PAYS=LO.ID_PAYS AND LO.ID_LANG=L.ID_LANG
WHERE L.LANGUE="ANGLAIS"
```

Résultat :

PAYS
CANADA
SUISSE

Exercice : Écrire une requête déterminant les langues officielles à Madagascar.

Corrigé :

```
-- LANGUES OFFICIELLES DE MADAGASCAR
SELECT LANGUE
FROM T_LANG AS L JOIN T_LANG_OFF AS LO
ON L.ID_LANG=LO.ID_LANG
JOIN T_PAYS AS P
ON LO.ID_PAYS=P.ID_PAYS
WHERE PAYS="MADAGASCAR"
```

ou

```
SELECT LANGUE
FROM T_LANG AS L JOIN T_LANG_OFF AS LO JOIN T_PAYS AS P
ON L.ID_LANG=LO.ID_LANG AND LO.ID_PAYS=P.ID_PAYS
WHERE PAYS="MADAGASCAR"
```

Exercice : Écrire une requête déterminant le nombre de langues officielles par continent.

Corrigé :

```
-- CONTINENT AVEC NB_LO
SELECT CONTINENT, COUNT(DISTINCT ID_LANG) AS NB_LO
FROM T_PAYS AS P JOIN T_LANG_OFF AS LO
ON P.ID_PAYS=LO.ID_PAYS
GROUP BY CONTINENT
```

Requête : Avec des jointures, on affiche la ou les langues communes entre Madagascar et le Canada.

```
-- LANGUES COMMUNES A MADAGASCAR ET AU CANADA
SELECT LANGUE FROM T_LANG L
JOIN T_LANG_OFF LO1 ON L.ID_LANG=LO1.ID_LANG
JOIN T_LANG_OFF LO2 ON LO2.ID_LANG=LO1.ID_LANG
JOIN T_PAYS P1 ON LO1.ID_PAYS=P1.ID_PAYS
JOIN T_PAYS P2 ON LO2.ID_PAYS=P2.ID_PAYS
WHERE P1.PAYS="CANADA" AND P2.PAYS="MADAGASCAR"
```

ou

```
SELECT LANGUE
FROM T_LANG L JOIN T_LANG_OFF LO1 JOIN T_LANG_OFF LO2
JOIN T_PAYS P1 JOIN T_PAYS P2
ON L.ID_LANG=LO1.ID_LANG AND LO2.ID_LANG=LO1.ID_LANG
AND LO1.ID_PAYS=P1.ID_PAYS AND LO2.ID_PAYS=P2.ID_PAYS
WHERE P1.PAYS="CANADA" AND P2.PAYS="MADAGASCAR"
```

On a effectué une *auto-jointure* en jointurant `T_LANG_OFF` sur lui-même, renommé respectivement en `L1` et `L2`. On peut éviter l'auto-jointure et procéder à une intersection :

```
-- LANGUES COMMUNES A MADAGASCAR ET AU CANADA
SELECT LANGUE FROM T_LANG
JOIN T_LANG_OFF ON T_LANG.ID_LANG=T_LANG_OFF.ID_LANG
JOIN T_PAYS ON  T_LANG_OFF.ID_PAYS =T_PAYS.ID_PAYS
WHERE PAYS="CANADA"
INTERSECT
SELECT LANGUE FROM T_LANG
JOIN T_LANG_OFF ON T_LANG.ID_LANG=T_LANG_OFF.ID_LANG
JOIN T_PAYS ON  T_LANG_OFF.ID_PAYS =T_PAYS.ID_PAYS
WHERE PAYS="MADAGASCAR"
```

ou

```
SELECT LANGUE
FROM T_LANG L JOIN T_LANG_OFF LO JOIN T_PAYS P
ON L.ID_LANG=LO.ID_LANG AND LO.ID_PAYS =P.ID_PAYS
WHERE PAYS="CANADA"
INTERSECT
SELECT LANGUE
FROM T_LANG L JOIN T_LANG_OFF LO JOIN T_PAYS P
ON L.ID_LANG=LO.ID_LANG AND LO.ID_PAYS =P.ID_PAYS
WHERE PAYS="MADAGASCAR"
```

Références

[1] SQL.sh : Cours et tutoriels sur le langage SQL, `https://sql.sh/`

[2] Developpez.com : Cours et tutoriels pour apprendre le SQL, `https://sql.developpez.com/`

[3] Thomas Nield, *Getting Started with SQL*, O'Reilly, 2016