

Cheatsheet MySQL

Table des matières

I. Conventions pour le nom et la structure de la base de données	3
II. Langage de manipulation de données	4
1 SELECT - Sélection de données simple ou dans une seule table	4
1.1 Requête SELECT simple (sans accès aux tables de la base de données)	4
1.2 Requête SELECT d'une seule table	5
1.3 La clause WHERE pour ajouter des conditions	5
1.4 La clause ORDER BY pour trier des résultats	5
1.5 La clause LIMIT pour afficher un sous-ensemble des enregistrements	5
1.6 Le <i>alias</i> AS pour renommer temporairement	6
1.6.1 Le <i>alias</i> AS pour renommer temporairement des champs ou calculs	6
1.6.2 Le <i>alias</i> AS pour renommer temporairement des tables	6
1.6.3 Le <i>alias</i> AS pour renommer temporairement des champs et des tables	6
1.7 Les fonctions	7
1.8 Les agrégations	8
2 SELECT - Sélection de données dans deux tables	9
2.1 La jointure CROSS JOIN (produit cartésien entre deux tables)	9
2.2 La jointure INNER JOIN	9
2.2.1 La jointure INNER JOIN simple	9
2.2.2 La jointure INNER JOIN avec le <i>alias</i> AS	10
2.2.3 La jointure INNER JOIN avec la clause WHERE	10
2.3 La jointure LEFT OUTER JOIN	11
3 SELECT - Sélection de données dans plus que deux tables	12
3.1 INNER JOIN multiple	12
3.1.1 Sélectionner toutes les correspondances	12
3.1.2 Sélectionner les correspondances pour un enregistrement	13
4 INSERT - Insertion des données dans une table	14
4.1 Requête INSERT INTO pour insérer un seul enregistrement	14
4.2 Requête INSERT INTO pour insérer plusieurs enregistrements à la fois	14
5 SELECT avec des sous-requêtes	14
6 UPDATE - Mise à jour de données d'une table	15
6.1 Requête UPDATE pour mettre à jour un seul enregistrement	15
6.2 Requête UPDATE pour mettre à jour plusieurs enregistrements	15
7 DELETE - Suppression de données d'une table	16
7.1 Requête DELETE pour effacer un seul enregistrement	16
7.2 Requête DELETE pour effacer plusieurs enregistrements	16
III. Langage de définition de données	17
8 CREATE - créer une structure de données	17

8.1	CREATE pour créer une table	17
8.1.1	Exemple	17
8.1.2	Instruction SQL	17
8.1.3	Dans l'interface phpMyAdmin	17
9	ALTER - modifier une structure de données	18
9.1	ALTER pour ajouter un index à un champ existant d'une table	18
9.1.1	Exemple	18
9.1.2	Instruction SQL	18
9.1.3	Dans l'interface phpMyAdmin	18
9.2	ALTER pour ajouter un champ à une table	18
9.2.1	Exemple	18
9.2.2	Instruction SQL	18
9.2.3	Dans l'interface phpMyAdmin	19
9.3	ALTER pour supprimer un champ à une table	19
9.3.1	Exemple	19
9.3.2	Instruction SQL	19
9.3.3	Dans l'interface phpMyAdmin	19
9.4	ALTER pour modifier un type de champs d'une table	19
9.4.1	Exemple	19
9.4.2	Instruction SQL	20
9.4.3	Dans l'interface phpMyAdmin	20
10	DROP - supprimer une structure de données	20
10.1	DROP pour supprimer une table (structure + données)	20
10.1.1	Exemple	20
10.1.2	Instruction SQL	20
11	RENAME - renommer une structure de données	20
11.1	RENAME pour renommer une table	20
11.1.1	Exemple	20
11.1.2	Instruction SQL	20
IV.	phpMyAdmin - La vue Concepteur	21
12	phpMyAdmin - La vue Concepteur	21
12.1	Déclarer une relation entre des tables	21
12.1.1	Exemple	21
12.1.2	Dans l'interface phpMyAdmin	21

I. Conventions pour le nom et la structure de la base de données

Pour des raisons pédagogiques et pour faciliter les corrections, je propose que vous respectiez des conventions suivantes pour la conception de votre base de données.

Rappelons-nous des 2 règles d'or pour la conception des bases de données :

- Pas de duplication de données.
- La structure d'une base de données ne doit pas évoluer avec le peuplement des données.

Structure de la base de données

Respectez les conventions suivantes :

- Le nom d'une table est à choisir en singulier, et tout comme les champs, sans caractères spéciaux, ni espace, ni accents, et tout en minuscules (seule exception : le caractère "souligné" ou en anglais "underscore" qui est le seul autorisé).
- Chaque table aura un champs `id` qui est clé primaire et mis en incrémentation automatique (A_I).
- Vous ne répétez jamais le nom de la table dans les noms des champs.
- Pour des chaînes de caractères, nous utilisons le type `TEXT` (qui s'adapte au nombre de caractères) et non pas `VARCHAR`.
- Les clés étrangères doivent avoir un `INDEX`, et ils s'appellent `id_xxx` où `xxx` est le nom de la table de la clé référencée. Si vous avez deux clés étrangères vers la même table (comme par exemple dans un message personnel), vous pouvez appeler les champs `id_xxx_emetteur` et `id_xxx_destinataire`.
- Chaque table peuplée par les utilisateurs aura un champs de type `DATETIME` (ou `TIMESTAMP` si vous préférez) qui s'appelle `datecreation`, et qui sera la date de création de l'enregistrement.

Dans l'exemple d'un site de petites annonces, les tables de la base de données pourraient être structurées comme suit, en respectant tous ces consignes :

Table **membre** (Données dynamiquement renseignées par vos utilisateurs)

id	pseudo	motdepasse	nom	prenom	datecreation
INT	TEXT	TEXT	TEXT	TEXT	DATETIME
1	xabi		Xabi	Lagarde	2022-03-21 18 :32 :31
2	skater		Patxi	Berthaud	2023-08-06 23 :22 :17

Table **categorie** (Données proposées par vous)

id	nom
INT	TEXT
1	Voitures
2	Immobilier

Table **annonce** (Données dynamiquement renseignées par vos utilisateurs)

id	titre	texte	prix	id_membre	id_categorie	valide	datecreation
INT	TEXT	TEXT	DECIMAL(10,2)	INT	INT	TINYINT	DATETIME
1	Location T1	...	465.00	1	2	1	2023-02-06 08 :14 :37
2	Twingo 2015	...	3400.00	2	1	0	2023-12-31 18 :31 :12

Exemple 1 :

id	nom	population	id_continent
INT	TEXT	INT	INT
1	Brésil	201103330	2
2	France	64768389	1
3	Suisse	7581000	1

id	nom
INT	TEXT
1	Europe
2	Amériques

Structure :

- Il y a deux **tables**, une table *pays* avec 4 **champs** (ou *attributes* ou *columns*), et une table *continent* avec 2 champs
- **Relation 1 : n** (ou *one-to-many*) : un *pays* appartient à un seul *continent*, et dans un *continent*, il peut y avoir plusieurs *pays*.

Données :

- 3 **enregistrements** (ou *lignes*) dans la table *pays*, et 2 **enregistrements** dans la table *continent*

II. Langage de manipulation de données

En SQL, un sous-ensemble des instructions est destiné à manipuler les données d'une base de données. Il y a typiquement quatre types de commandes SQL de manipulation de données :

- SELECT - sélection de données dans une table
- INSERT - insertion de données dans une table
- DELETE - suppression de données d'une table
- UPDATE - mise à jour de données d'une table

1 SELECT - Sélection de données simple ou dans une seule table

Syntaxe :

SELECT $c_1, \dots, c_p$ **FROM** $t_1, \dots, t_n$ **WHERE** b_1 **GROUP BY** $c_{i1}, c_{i2}, \dots$ **HAVING** b_2
ORDER BY $c_{j1}, c_{j2}, \dots$ **LIMIT** ...

- c_i : le ou les champs ou calculs, représentant le résultat attendu, renommé avec AS si besoin
- t_i : la ou les tables
- b_1, b_2 : condition booléenne sur les c_i

Remarque : L'ordre des mots réservés doit être respecté, mais certaines clauses sont optionnelles (grisées).

1.1 Requête SELECT simple (sans accès aux tables de la base de données)

SELECT 5+6;	Sélectionne le résultat du calcul de 5+6
SELECT 'Hello World';	Sélectionne la chaîne de caractères <i>Hello World</i>
SELECT 5+6, 'Hello World';	Sélectionne le résultat du calcul de 5+6 et la chaîne de caractères <i>Hello World</i>

Remarque : Les chaînes de caractères doivent être limitées par des simples quotes '.

1.2 Requête SELECT d'une seule table

SELECT * FROM pays;	Sélectionne tous les champs de la table <i>pays</i>
SELECT * FROM sgbd.pays;	Même requête, en spécifiant explicitement le nom de la base de données, p.ex. <i>sgbd</i> , où se trouve la table.
SELECT nom FROM pays;	Sélectionne le champ <i>nom</i> de la table <i>pays</i>
SELECT `nom` FROM `pays`;	Même requête. Voir la remarque ci-dessous.
SELECT pays.nom FROM pays;	Même requête, en répétant le nom de la table (utile pour des requêtes dans plusieurs tables, voir Section 2).
SELECT id, nom FROM pays;	Sélectionne les champs <i>id</i> , <i>nom</i> de la table <i>pays</i>
SELECT DISTINCT id_continent FROM pays;	Sélectionne les identifiants des continents de la table <i>pays</i> en sélectionnant uniquement des résultats distincts, donc sans redondances

Remarque : Le ` permet de délimiter des noms des champs et des tables pour les mettre des espaces, caractères spéciaux, ou mots réservés SQL. **Je déconseille** : pour une meilleure lisibilité, utilisez les noms simples, comme ça vous n'aurez pas besoin de `.

1.3 La clause WHERE pour ajouter des conditions

SELECT * FROM pays WHERE nom='France';	Sélectionne tous les champs de la table <i>pays</i> dont le champ <i>nom</i> correspond à France
SELECT * FROM pays WHERE nom LIKE 'F%';	Sélectionne tous les champs de la table <i>pays</i> dont le champ <i>nom</i> commence avec F ou f
SELECT * FROM pays WHERE nom LIKE '%uis%';	Sélectionne tous les champs de la table <i>pays</i> qui contient la chaîne de caractère <i>uis</i>
SELECT * FROM pays WHERE nom='France' AND id<10;	Opérations booléennes : Sélectionne tous les champs de la table <i>pays</i> dont le champ <i>nom</i> correspond à France et dont le champ <i>id</i> est strictement inférieur à 10
SELECT * FROM pays WHERE id IN (1,2,4)	Sélectionne tous les champs de la table <i>pays</i> dont le champ <i>id</i> est 1, 2, ou 4.

1.4 La clause ORDER BY pour trier des résultats

SELECT * FROM pays ORDER BY nom;	Sélectionne tous les champs de la table <i>pays</i> trié par <i>nom</i> , par ordre croissant
SELECT * FROM pays ORDER BY nom ASC;	Sélectionne tous les champs de la table <i>pays</i> trié par <i>nom</i> , par ordre croissant
SELECT * FROM pays ORDER BY nom DESC;	Sélectionne tous les champs de la table <i>pays</i> trié par <i>nom</i> , par ordre décroissant
SELECT * FROM pays ORDER BY id_continent ASC, nom DESC;	Sélectionne tous les champs de la table <i>pays</i> trié d'abord par <i>id_continent</i> par ordre croissant, et ensuite par <i>nom</i> , par ordre décroissant

1.5 La clause LIMIT pour afficher un sous-ensemble des enregistrements

SELECT * FROM pays ORDER BY population ASC LIMIT 5;	Sélectionne tous les champs de la table <i>pays</i> , trié par <i>population</i> , par ordre décroissant, mais seulement les 5 premiers enregistrements
SELECT * FROM pays ORDER BY population ASC LIMIT 10,5;	Sélectionne tous les champs de la table <i>pays</i> , trié par <i>population</i> , par ordre décroissant, mais seulement 5 enregistrements à partir du 10ème enregistrement
SELECT * FROM pays WHERE id_continent=1 ORDER BY population ASC LIMIT 10,5;	Sélectionne tous les champs de la table <i>pays</i> dont le champ <i>id_continent</i> est égal à 1, trié par <i>population</i> , par ordre décroissant, mais seulement 5 enregistrements à partir du 10ème enregistrement

1.6 Le *alias* AS pour renommer temporairement

Les **alias** permettent de renommer temporairement un calcul, un champs ou une table dans une requête. Cela permet dans certains cas d'augmenter la lisibilité, notamment lorsqu'on travaillera avec des fonction, agrégations, ou plusieurs tables.

1.6.1 Le *alias* AS pour renommer temporairement des champs ou calculs

Exemples :

<pre>SELECT 5+6 AS resultat;</pre>	Sélectionne le résultat du calcul de 5+6 et le renomme en <i>resultat</i>
<pre>SELECT 5+6 AS resultat, 'Hello World' AS chaine;</pre>	Alias pour plusieurs champs : Sélectionne le résultat du calcul de 5+6 et la chaîne de caractères <i>Hello World</i> , et les renomme en <i>resultat</i> et <i>chaine</i>
<pre>SELECT population AS pop FROM pays;</pre>	Alias pour un champ : sélectionne le champ <i>population</i> de la table <i>pays</i> et le renomme en <i>pop</i> pour ce résultat de requête
<pre>SELECT id, population AS pop FROM pays;</pre>	Alias pour un champ : Sélectionne le champ <i>id</i> et <i>population</i> de la table <i>pays</i> et renomme ce dernier en <i>pop</i> pour cette requête

1.6.2 Le *alias* AS pour renommer temporairement des tables

Exemples :

<pre>SELECT p.id, p.nom FROM pays AS p;</pre>	Renomme temporairement la table <i>pays</i> en <i>p</i> et sélectionne les champs <i>id</i> et <i>nom</i>
<pre>SELECT p.id, p.nom FROM pays p;</pre>	Même chose, en forme courte sans AS
<pre>SELECT p.* FROM pays AS p;</pre>	Renomme temporairement la table <i>pays</i> en <i>p</i> et sélectionne tous les champs

1.6.3 Le *alias* AS pour renommer temporairement des champs et des tables

Exemple :

<pre>SELECT p.id AS pays_id, p.nom AS pays_nom FROM pays p;</pre>	Alias pour une table : Sélectionne les champs de la table <i>pays</i> , et renomme la table en <i>p</i> , et renomme les champs en <i>pays_id</i> , et <i>pays_nom</i> pour cette requête
---	---

Résultat pour cet exemple :

pays_id	pays_nom
1	Brésil
2	France
3	Suisse

1.7 Les fonctions

Les fonctions permettent de manipuler les données. Il existe plusieurs familles de fonctions :

- des fonctions mathématiques (SQRT, ROUND, FLOOR, ABS, SIN, RAND, ...),
- des fonctions de manipulation de dates (NOW(), DATEDIFF, DAY, MONTH, YEAR, ...)
- des fonctions de manipulation de caractères (SUBSTR, SUBSTR, UPPER, LOWER, ...)

Exemples pour des fonctions mathématiques :

SELECT SQRT(2);	Sélectionne le résultat de la racine carré de 2
SELECT SQRT(2) AS sqrt2;	Sélectionne le résultat de la racine carré de 2 et le renomme en <i>sqrt2</i> pour cette requête
SELECT nom FROM pays ORDER BY RAND() ASC;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et trie les enregistrement aléatoirement

Exemples pour des fonctions de manipulation de dates :

SELECT NOW();	Sélectionne la date et l'heure de maintenant
SELECT NOW() AS maintenant;	Sélectionne la date et l'heure de maintenant et le renomme en <i>maintenant</i> pour cette requête
SELECT DATEDIFF(NOW(), '2019-12-31') AS jours_depuis_2019	Calcule le nombre de jours entre aujourd'hui et le 31 décembre 2019 et le renomme en <i>jours_depuis_2019</i> pour cette requête

Exemples pour des fonctions de manipulation de caractères :

SELECT UPPER(nom) FROM pays;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et applique la fonction <i>UPPER</i> pour le transformer en tout majuscule
SELECT LOWER(nom) AS nom_miniscule FROM pays;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et applique la fonction <i>LOWER</i> pour le transformer en tout minuscule et le renomme en <i>nom_miniscule</i> pour cette requête
SELECT LENGTH(nom) FROM pays;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et applique la fonction <i>LENGTH</i> pour déterminer la longueur en nombre de caractères
SELECT SUBSTR(nom, 0, 2) FROM pays;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et applique la fonction <i>SUBSTR</i> pour sélectionner les deux premiers caractères
SELECT MD5(nom) FROM pays;	Sélectionne le champs <i>nom</i> de la table <i>pays</i> et applique la fonction MD5

1.8 Les agrégations

Les agrégations se réalisent à travers des fonctions qui permettent le regroupement de plusieurs tuples en vue d'un traitement. Les plus connus sont COUNT, MIN, MAX, AVG, SUM, ...

Agrégations renvoyant un seul enregistrement de la table

Exemples :

SELECT COUNT(id) FROM pays;	Sélectionne le nombre d'enregistrements de de la table <i>pays</i>
SELECT COUNT(id) AS nb_pays FROM pays;	Sélectionne le nombre d'enregistrements de de la table <i>pays</i> et le renomme en <i>nb_pays</i> pour cette requête
SELECT COUNT(DISTINCT id_continent) FROM pays;	Sélectionne le nombre de champs <i>id_continent</i> sans doublons de la table <i>pays</i>
SELECT MIN(population), MAX(population) FROM pays;	Sélectionne la plus grande et la plus petite valeur du champ <i>population</i> de de la table <i>pays</i>
SELECT AVG(population) FROM pays;	Sélectionne la moyenne des valeurs du champ <i>population</i> de de la table <i>pays</i>
SELECT SUM(population) FROM pays;	Sélectionne la somme des valeurs du champ <i>population</i> de de la table <i>pays</i>

Remarque : Il ne faut pas mélanger des valeurs agrégés et non-agrégés. Par exemple, dans la requête **SELECT MIN(population), id FROM pays**, l'identifiant *id* ne correspond pas forcément au pays ayant la plus faible population ...

Agrégations avec la clause GROUP BY, pouvant renvoyer plusieurs enregistrements

Exemples :

SELECT COUNT(id), SUM(population), id_continent FROM pays GROUP BY id_continent	Sélectionne le nombre d'enregistrements par <i>id_continent</i> , la somme des population par continent, et <i>id_continent</i> de la table <i>pays</i>
SELECT COUNT(id), SUM(population) AS somme, id_continent FROM pays GROUP BY id_continent ORDER BY somme DESC	Même requête, en renommant temporairement la somme des <i>populations</i> par <i>continent</i> en somme , et en triant le résultat en commençant par la plus grande somme des <i>populations</i>
SELECT COUNT(id), SUM(population) AS somme, id_continent FROM pays WHERE id>1 GROUP BY id_continent ORDER BY somme DESC	Même requête, mais prenant en compte uniquement les pays ayant des identifiants strictement supérieur à 1

Remarque : Le seul champ non-agrégé permis est le champs qui fait objet du groupement (ici, *id_continent*).

Agrégations avec la clause GROUP BY et la clause HAVING

Exemples :

SELECT COUNT(id) AS nb_pays_par_continent, id_continent FROM pays GROUP BY id_continent HAVING nb_pays_par_continent>=2	Sélectionne le nombre d'enregistrements par <i>id_continent</i> et les renomme en <i>nb_pays_par_continent</i> et <i>id_continent</i> de la table <i>pays</i> , mais uniquement pour les <i>continents</i> ayant au moins 2 <i>pays</i>
SELECT COUNT(id) AS nb_pays_par_continent, id_continent FROM pays WHERE population >= 10000000 GROUP BY id_continent HAVING nb_pays_par_continent>=2	Même requête, mais prenant en compte uniquement les pays avec une population supérieur à 10 millions. Notez que pour les données de l'exemple 1, aucun résultat est renvoyé.

Remarque : **Différence entre HAVING et WHERE** : Les conditions de HAVING s'appliquent après le groupement, les conditions de WHERE forment le groupement.

2 SELECT - Sélection de données dans deux tables

Pour sélectionner des données dans deux tables, nous allons faire des requêtes avec des jointures entre des tables. En MySQL, il existe 3 types de jointure :

- La jointure CROSS JOIN (produit cartésien entre les deux tables)
- La jointure INNER JOIN (la plus fréquente)
- La jointure LEFT OUTER JOIN

2.1 La jointure CROSS JOIN (produit cartésien entre deux tables)

Les deux instructions suivantes sont équivalentes :

<pre>SELECT pays.*, continent.* FROM pays, continent;</pre>	Sélectionne tous les champs des deux tables du produit cartésien des deux tables <i>pays</i> et <i>continent</i> (aussi appelé CROSS JOIN). Si les deux tables ont <i>m</i> et <i>n</i> enregistrements, la requête retourne $n * m$ enregistrements.
<pre>SELECT pays.*, continent.* FROM pays CROSS JOIN continent;</pre>	Version CROSS JOIN.

id	nom	population	id_continent	id	nom
1	Brésil	201103330	2	1	Europe
1	Brésil	201103330	2	2	Amériques
2	France	64768389	1	1	Europe
2	France	64768389	1	2	Amériques
3	Suisse	7581000	1	1	Europe
3	Suisse	7581000	1	2	Amériques

2.2 La jointure INNER JOIN

2.2.1 La jointure INNER JOIN simple

Les deux instructions suivantes sont équivalentes :

<pre>SELECT pays.*, continent.* FROM pays, continent WHERE pays.id_continent = continent.id</pre>	Sélectionne tous les champs des deux tables <i>pays</i> et <i>continent</i> dont la clé étrangère <i>pays.id_continent</i> correspond à la clé primaire <i>continent.id</i> .
<pre>SELECT * FROM pays INNER JOIN continent ON pays.id_continent = continent.id</pre>	Version "INNER JOIN". INNER JOIN et JOIN sont équivalentes.

id	nom	population	id_continent	id	nom
1	Brésil	201103330	2	2	Amériques
2	France	64768389	1	1	Europe
3	Suisse	7581000	1	1	Europe

2.2.2 La jointure INNER JOIN avec le *alias* AS

Comme certains champs ont le même nom, l'accès peut être ambigu. Pour cela, utilisez un ALIAS comme suit :

Les deux instructions suivantes avec le *alias* AS sont équivalentes :

<pre>SELECT pays.id AS pays_id, pays.nom AS pays_nom, population, id_continent, continent.id AS continent_id, continent.nom AS continent_nom FROM pays, continent WHERE pays.id_continent = continent.id</pre>	Sélectionne les champs des deux tables <i>pays</i> et <i>continent</i> dont la clé étrangère <i>pays.id_continent</i> correspond à la clé primaire <i>continent.id</i> , et renomme les champs ambigus <i>id</i> et <i>nom</i> des tables <i>pays</i> et <i>continent</i> , en <i>pays_id</i> et <i>pays_nom</i> , respectivement, <i>continent_id</i> et <i>continent_nom</i> , pour cette requête.
<pre>SELECT pays.id AS pays_id, pays.nom AS pays_nom, population, id_continent, continent.id AS continent_id, continent.nom AS continent_nom FROM pays INNER JOIN continent ON pays.id_continent = continent.id</pre>	Version "INNER JOIN"

pays_id	pays_nom	population	id_continent	continent_id	continent_nom
1	Brésil	201103330	2	2	Amériques
2	France	64768389	1	1	Europe
3	Suisse	7581000	1	1	Europe

2.2.3 La jointure INNER JOIN avec la clause WHERE

Les deux instructions suivantes avec ALIAS sont équivalentes :

<pre>SELECT pays.id AS pays_id, pays.nom AS pays_nom, population, continent.id AS continent_id, continent.nom AS continent_nom FROM pays, continent WHERE pays.id_continent = continent.id AND pays.nom LIKE 'France';</pre>	Pour le pays <i>France</i> , sélectionner les champs des deux tables <i>pays</i> et <i>continent</i> dont la clé étrangère <i>pays.id_continent</i> correspond à la clé primaire <i>continent.id</i> , et renomme les champs ambigus <i>id</i> et <i>nom</i> des tables <i>pays</i> et <i>continent</i> , en <i>pays_id</i> et <i>pays_nom</i> , respectivement, <i>continent_id</i> et <i>continent_nom</i> , pour cette requête.
<pre>SELECT pays.id AS pays_id, pays.nom AS pays_nom, population, continent.id AS continent_id, continent.nom AS continent_nom FROM pays INNER JOIN continent ON pays.id_continent = continent.id WHERE pays.nom LIKE 'France';</pre>	Version "INNER JOIN"

pays_id	pays_nom	population	continent_id	continent_nom
2	France	64768389	1	Europe

2.3 La jointure LEFT OUTER JOIN

Parfois, il existe des enregistrements avec des clés étrangères qui ne sont pas saisis.

La jointure LEFT OUTER JOIN permet d'inclure ces enregistrements dans les résultats.

```
SELECT * FROM pays
LEFT OUTER JOIN continent
ON pays.id_continent = continent.id
```

En plus des résultats de INNER JOIN, LEFT OUTER JOIN ressort aussi les enregistrement de la table *pays* dont la clé étrangère *pays.id_continent* a la valeur NULL. Pour ces enregistrements, la valeur des champs de la table *continent* sera NULL. Remarque : LEFT OUTER JOIN et LEFT JOIN sont équivalentes.

Résultat pour un exemple où la table *pays* contient un enregistrement "Groenland", avec une population de 56025 et une clé étrangère *id_continent* avec la valeur NULL :

id	nom	population	id_continent	id	nom
1	Brésil	201103330	2	2	Amériques
2	France	64768389	1	1	Europe
3	Suisse	7581000	1	1	Europe
4	Groenland	56025	NULL	NULL	NULL

Remarque 1 : En MySQL, il n'y a pas de jointure RIGHT OUTER JOIN - de toute façon, ces requêtes peuvent être converties en une jointure LEFT OUTER JOIN en inversant les noms des tables entre FROM et LEFT OUTER JOIN.

Remarque 2 : En MySQL, il n'y a pas non plus de jointure FULL OUTER JOIN - on pourrait construire cette jointure en faisant deux jointures LEFT OUTER JOIN et en fusionnant le résultat avec UNION.

Exemple 2 :

Table nouvelle			Table tag		Table nouvelle_tag		
id	texte	date	id	nom	id	id_nouvelle	id_tag
INT	TEXT	DATETIME	INT	TEXT	INT	INT	INT
1	L'AS Saint Etienne en route vers la Ligue 1	2024-04-02 08 :32 :31	1	Sport	1	1	1
2	Encore un James Bond avec Daniel Craig ?	2023-11-26 11 :03 :22	2	Foot	2	1	2
3	Droit à une feuille manuscrite à l'examen	2024-04-01 11 :23 :01	3	Cinéma	3	2	3
4	On est à 4 mois des JO 2024	2024-03-26 17 :52 :40	4	Université	4	2	5
			5	People	5	3	4
					6	4	1

- Structure :
- Il y a trois **tables**, une table *nouvelle* avec 3 **champs** (ou *attributes* ou *columns*), une table *tag* avec 2 champs, et une **table de jointure** *nouvelle_tag* avec 3 champs, dont 1 clé primaire et 2 clés étrangères
 - Relation n :m** (ou *many-to-many*) : une *nouvelle* peut avoir plusieurs *tags*, et un *tag* peut être associé à plusieurs *nouvelles*.
- Données :
- 4 **enregistrements** dans la table *nouvelle*, 5 **enregistrements** dans la table *tag*, et 6 **enregistrements** dans la table de jointure *nouvelle_tag*

3 SELECT - Sélection de données dans plus que deux tables

3.1 INNER JOIN multiple

3.1.1 Sélectionner toutes les correspondances

Les deux instructions suivantes sont équivalentes :

SELECT nouvelle.*, tag.* FROM nouvelle, nouvelle_tag, tag WHERE nouvelle_tag.id_nouvelle = nouvelle.id AND nouvelle_tag.id_tag = tag.id	Sélectionne l'identifiant et le texte de la table <i>nouvelle</i> , le nom de la table <i>tag</i> , qui sont liés par la table de jointure <i>nouvelle_tag</i>
SELECT nouvelle.*, tag.* FROM nouvelle INNER JOIN nouvelle_tag ON nouvelle_tag.id_nouvelle = nouvelle.id INNER JOIN tag ON nouvelle_tag.id_tag = tag.id	Version "INNER JOIN"

id	texte	date	id	nom
1	L'AS Saint Etienne en route vers la Ligue 1	2024-04-02 08 :32 :31	1	Sport
1	L'AS Saint Etienne en route vers la Ligue 1	2024-04-02 08 :32 :31	2	Foot
2	Encore un James Bond avec Daniel Craig ?	2023-11-26 11 :03 :22	3	Cinéma
2	Encore un James Bond avec Daniel Craig ?	2023-11-26 11 :03 :22	5	People
3	Droit à une feuille manuscrite à l'examen	2024-04-01 11 :23 :01	4	Sortie de film
4	On est à 4 mois des JO 2024	2024-03-26 17 :52 :40	1	Sport

3.1.2 Sélectionner les correspondances pour un enregistrement

Exemple : Tous les *tags* pour la dépêche avec l'identifiant *id=1*

Les deux instructions suivantes sont équivalentes :

<pre>SELECT nouvelle.id, nouvelle.texte, nouvelle_tag.id_tag, tag.nom FROM nouvelle, nouvelle_tag, tag WHERE nouvelle_tag.id_nouvelle = nouvelle.id AND nouvelle_tag.id_tag = tag.id AND nouvelle.id=1;</pre>	Sélectionne tous les champs des deux tables <i>pays</i> et <i>continent</i> dont la clé étrangère pays.id_continent correspond à la clé primaire continent.id .
<pre>SELECT nouvelle.id, nouvelle.texte, nouvelle_tag.id_tag, tag.nom FROM nouvelle INNER JOIN nouvelle_tag ON nouvelle_tag.id_nouvelle = nouvelle.id INNER JOIN tag ON nouvelle_tag.id_tag = tag.id AND nouvelle.id=1;</pre>	Version "INNER JOIN"

id	texte	id_tag	nom
1	L'AS Saint Etienne en route vers la Ligue 1	1	Sport
1	L'AS Saint Etienne en route vers la Ligue 1	2	Foot

Exemple : Toutes les *nouvelles* ayant le tag *Sport*

Les deux instructions suivantes sont équivalentes :

<pre>SELECT nouvelle.id, nouvelle.texte, nouvelle_tag.id_tag, tag.nom FROM nouvelle, nouvelle_tag, tag WHERE nouvelle_tag.id_nouvelle = nouvelle.id AND nouvelle_tag.id_tag = tag.id AND tag.nom LIKE 'Sport';</pre>	Sélectionne tous les champs des deux tables <i>pays</i> et <i>continent</i> dont la clé étrangère pays.id_continent correspond à la clé primaire continent.id .
<pre>SELECT nouvelle.id, nouvelle.texte, nouvelle_tag.id_tag, tag.nom FROM nouvelle INNER JOIN nouvelle_tag ON nouvelle_tag.id_nouvelle = nouvelle.id INNER JOIN tag ON nouvelle_tag.id_tag = tag.id AND tag.nom LIKE 'Sport';</pre>	Version "INNER JOIN"

id	texte	id_tag	nom
1	L'AS Saint Etienne en route vers la Ligue 1	1	Sport
4	On est à 4 mois des JO 2024	1	Sport

4 INSERT - Insertion des données dans une table

Syntaxe :

INSERT INTO $t(c_1, \dots, c_n)$ **VALUES** $(v_1, \dots, v_n)$, $(v_1, \dots, v_n)$, ...

- t : le nom de la table
- c_i : le ou les noms des champs
- v_i : le ou les valeurs à mettre dans les champs

Remarque : Les champs en incrémentation automatique ainsi que ceux ayant une valeur par défaut peuvent être omis.

4.1 Requête INSERT INTO pour insérer un seul enregistrement

Exemples pour insérer un seul enregistrement

INSERT INTO nouvelle (texte, date) VALUES ('Il fera beau ce lundi', '2020-04-13 08:06:05');	Insère dans la table <i>nouvelle</i> un nouvel enregistrement avec des valeurs pour les deux champs <i>texte</i> , et <i>date</i> . Le champ <i>id</i> en incrémentation automatique n'est pas spécifié - cet identifiant est attribué automatiquement.
INSERT INTO nouvelle (texte, date) VALUES ('Il fera beau ce lundi', NOW());	Même requête, avec la fonction NOW() pour utiliser la date et l'heure actuelle pour le champs <i>date</i> .
INSERT INTO nouvelle (texte, date) VALUES ('Il y a moins d'accidents cette année.', NOW());	Même requête, avec un texte qui nécessite l'échappement du caractère ' pour qu'il ne soit pas interprété comme fin de chaîne de caractère
INSERT INTO nouvelle (id, texte, date) VALUES (NULL, 'Il fera beau ce lundi', NOW());	Même requête, en spécifiant NULL pour le champs en incrémentation automatique, ayant le même effet.
INSERT INTO sgbd.nouvelle (id, texte, date) VALUES (NULL, 'Il fera beau ce lundi', NOW());	Même requête, en spécifiant explicitement le nom de la base de données, p.ex. <i>sgbd</i> , où se trouve la table.

4.2 Requête INSERT INTO pour insérer plusieurs enregistrements à la fois

Exemples pour insérer plusieurs enregistrements à la fois

INSERT INTO nouvelle (texte, date) VALUES ('Il fera beau ce lundi', NOW()), ('Il y a moins d'accidents cette année.', NOW());	Insère dans la table <i>nouvelle</i> un nouvel enregistrement avec des valeurs pour les trois champs <i>id</i> , <i>texte</i> , et <i>date</i> .
---	--

5 SELECT avec des sous-requêtes

à venir

6 UPDATE - Mise à jour de données d'une table

Syntaxe :

UPDATE *t* **SET** $c_1 = v_1, c_2 = v_2, \dots$ **WHERE** b_1

- t : le nom de la table
- c_i : le ou les noms des champs
- v_i : le ou les valeurs à mettre dans les champs
- b_1 : condition booléenne

6.1 Requête UPDATE pour mettre à jour un seul enregistrement

Exemples pour mettre à jour un seul enregistrement

UPDATE nouvelle SET texte = 'Il fera mauvais ce lundi' WHERE id=5;	Met à jour la table <i>nouvelle</i> en affectant le champ <i>texte</i> avec une nouvelle valeur à l'enregistrement avec l'identifiant <i>id=5</i> .
UPDATE nouvelle SET texte = 'Il fera mauvais ce lundi', date = NOW() WHERE id=5;	Met à jour la table <i>nouvelle</i> en affectant les champs <i>texte</i> et <i>date</i> avec des nouvelles valeurs à l'enregistrement avec l'identifiant <i>id=5</i> .

6.2 Requête UPDATE pour mettre à jour plusieurs enregistrements

Exemples pour mettre à jour plusieurs enregistrements

UPDATE nouvelle SET date = '2020-04-13 08:06:05' WHERE id<5;	Met à jour la table <i>nouvelle</i> en affectant le champ <i>date</i> avec une nouvelle valeur pour tous les enregistrements avec les identifiants <i>id<5</i> .
UPDATE nouvelle SET date = '2020-04-13 08:06:05'	Met à jour la table <i>nouvelle</i> en affectant le champ <i>date</i> avec une nouvelle valeur pour tous les enregistrements.

7 DELETE - Suppression de données d'une table

Syntaxe :

DELETE FROM *t* **WHERE** *b₁*

- *t* : le nom de la table
- *b₁* : condition booléenne

7.1 Requête DELETE pour effacer un seul enregistrement

DELETE FROM nouvelle WHERE id=5;	Supprime tous les enregistrements de la table <i>nouvelle</i> avec l'identifiant <i>id=5</i> .
----------------------------------	--

7.2 Requête DELETE pour effacer plusieurs enregistrements

DELETE FROM nouvelle WHERE id<5;	Supprime tous les enregistrements de la table <i>nouvelle</i> avec l'identifiant <i>id<5</i> .
DELETE FROM nouvelle;	Supprime tous les enregistrements de la table <i>nouvelle</i> .

III. Langage de définition de données

En MySQL, il existe quatre types de commandes de définition de données :

- CREATE - créer une structure de données
- ALTER - modifier une structure de données
- DROP - supprimer une structure de données
- RENAME - renommer une structure de données

8 CREATE - créer une structure de données

8.1 CREATE pour créer une table

8.1.1 Exemple

Exemple : création d'une table pays avec les 4 champs suivants :

- un champ *id* de type INT, qui est **clé primaire**, est mis en **incrément automatique**
- un champ *nom* de type TEXT
- un champ *population* de type INT
- un champ *id_continent* de type INT, qui est une **clé étrangère**. Les clés étrangères doivent être indexées avec INDEX pour des raisons d'efficacité

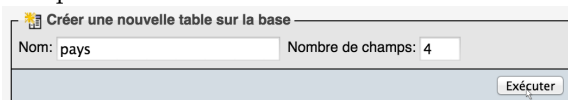
8.1.2 Instruction SQL

```
CREATE TABLE pays
(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  nom TEXT NOT NULL ,
  population INT NOT NULL ,
  id_continent INT NOT NULL ,

  INDEX ( id_continent )
) ENGINE = INNODB
```

8.1.3 Dans l'interface phpMyAdmin

Sélectionner d'abord votre base de données, et ensuite déterminer le nom de votre table et le nombre de champs, et cliquez sur "Exécuter" :



Puis renseignez les noms des champs, leurs types, et leur INDEX le cas échéant, et cliquez sur "Sauvegarder".

Nom de la table: Ajouter 1 colonne(s) Exécuter

Nom	Type	Taille/Valeurs*	Défaut	Interclassement	Attributs	Null	Index	A I	Comr
id	INT		Aucune			☐	PRIMARY	☒	
nom	TEXT		Aucune			☐	---	☐	
population	INT		Aucune			☐	---	☐	
id_continent	INT		Aucune			☐	INDEX	☐	

Commentaires sur la table :

Moteur de stockage :

Interclassement :

Définition de PARTITION :

Sauvegarder

9 ALTER - modifier une structure de données

9.1 ALTER pour ajouter un index à un champ existant d'une table

Cette opération est nécessaire avant de lier une clé étrangère à une clé primaire d'une autre table : il faut mettre un INDEX à la clé étrangère.

9.1.1 Exemple

Exemple : ajouter un INDEX au champ *id_continent* de la table *pays*

9.1.2 Instruction SQL

```
ALTER TABLE pays ADD INDEX ( id_continent )
```

9.1.3 Dans l'interface phpMyAdmin

Sélectionner d'abord l'onglet "Structure", et ensuite cliquez sur l'icône de "Index" du champs à modifier :

Afficher Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Suivi Déclencheurs

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
☐ 1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index plus
☐ 2	nom	text	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index plus
☐ 3	population	int(11)			Non	Aucune		Modifier Supprimer Primaire Unique Index plus
☐ 4	id_continent	int(11)			Oui	NULL		Modifier Supprimer Primaire Unique Index plus

Tout cocher Pour la sélection : Afficher Modifier Supprimer Primaire Unique Index

9.2 ALTER pour ajouter un champ à une table

9.2.1 Exemple

Exemple : ajouter le champ *superficie* de type FLOAT à la table *pays*, après le champ *population*

9.2.2 Instruction SQL

```
ALTER TABLE pays ADD superficie FLOAT NOT NULL AFTER population;
```

9.2.3 Dans l'interface phpMyAdmin

Sélectionner d'abord l'onglet "Structure", et ensuite sélectionnez dans la section "Ajouter" à quel endroit vous voulez ajouter le champs, et cliquer sur "Exécuter" :

Structure

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer plus
2	nom	text	latin1_swedish_ci		Non	Aucune		Modifier Supprimer plus
3	population	int(11)			Non	Aucune		Modifier Supprimer plus
4	id_continent	int(11)			Oui	NULL		Modifier Supprimer plus

Pour la sélection : Afficher Modifier Supprimer Primaire Unique Index

Version imprimable Vue relationnelle Suggérer des optimisations de structure Suivre la table Déplacer des colonnes

Ajouter 1 colonne(s) En fin de table En début de table Après population Exécuter

Ensuite, ajoutez le champ *superficie* de type FLOAT et cliquez sur "Sauvegarder".

Structure

Nom	Type	Taille/Valeurs*	Défaut	Interclassement	Attributs	Null
superficie	FLOAT		Aucune			

Sauvegarder

9.3 ALTER pour supprimer un champ à une table

9.3.1 Exemple

Exemple : supprimez le champ *superficie* de la table *pays*

9.3.2 Instruction SQL

```
ALTER TABLE pays DROP superficie;
```

9.3.3 Dans l'interface phpMyAdmin

Sélectionner d'abord l'onglet "Structure", et ensuite cliquez sur "Modifier" du champs à modifier, et confirmez ensuite :

Structure

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer plus
2	nom	text	latin1_swedish_ci		Non	Aucune		Modifier Supprimer plus
3	population	int(11)			Non	Aucune		Modifier Supprimer plus
4	superficie	float			Non	Aucune		Modifier Supprimer plus
5	id_continent	int(11)			Oui	NULL		Modifier Supprimer plus

Pour la sélection : Afficher Modifier Supprimer Primaire Unique Index

9.4 ALTER pour modifier un type de champs d'une table

9.4.1 Exemple

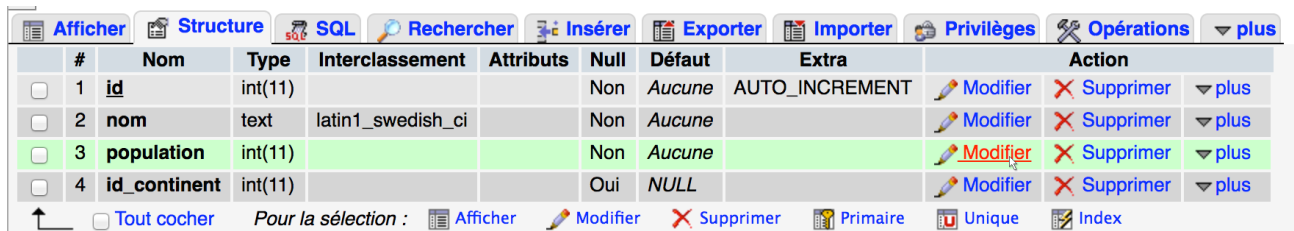
Exemple : modifier le type de champ *poupulation* de la table *pays* en BIGINT

9.4.2 Instruction SQL

```
ALTER TABLE pays CHANGE population population BIGINT(11) NOT NULL;
```

9.4.3 Dans l'interface phpMyAdmin

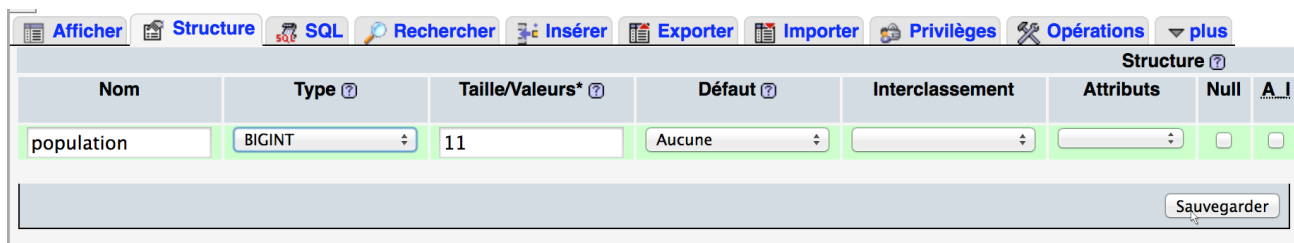
Sélectionner d'abord l'onglet "Structure", et ensuite cliquez sur "Modifier" du champs à modifier :



	#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
☐	1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer plus
☐	2	nom	text	latin1_swedish_ci		Non	Aucune		Modifier Supprimer plus
☒	3	population	int(11)			Non	Aucune		Modifier Supprimer plus
☐	4	id_continent	int(11)			Oui	NULL		Modifier Supprimer plus

☐ Tout cocher Pour la sélection : [Afficher](#) [Modifier](#) [Supprimer](#) [Primaire](#) [Unique](#) [Index](#)

Ensuite, modifiez le type de champs en BIGINT et cliquez sur "Sauvegarder".



Nom	Type ?	Taille/Valeurs* ?	Défaut ?	Interclassement	Attributs	Null	A !
population	BIGINT	11	Aucune			☐	☐

[Sauvegarder](#)

10 DROP - supprimer une structure de données

10.1 DROP pour supprimer une table (structure + données)

10.1.1 Exemple

Exemple : supprimer la table *pays*

10.1.2 Instruction SQL

```
DROP TABLE pays
```

11 RENAME - renommer une structure de données

11.1 RENAME pour renommer une table

11.1.1 Exemple

Exemple : renommer la table *pays* en *projet_pays*

11.1.2 Instruction SQL

```
RENAME TABLE pays TO projet_pays
```

IV. phpMyAdmin - La vue Concepteur

12 phpMyAdmin - La vue Concepteur

12.1 Déclarer une relation entre des tables

Pour déclarer une relation entre tables, c'est à dire lier une clé étrangère d'une table à une clé étrangère d'une autre table, veillez à ce que les conditions suivantes soit remplies :

- La clé étrangère doit être indexé! (cf. Section 9.1)
- Les données doivent exister, c'est à dire si la clé étrangère fait référence à des identifiants qui correspondent à des données d'un champ de clé primaire d'une autre table, les données pour ces identifiants doivent exister.
Exemple : si un pays fait référence à *id_continent* 2, le continent avec l'identifiant *id*=2 doit exister!

12.1.1 Exemple

Exemple : relier la table *pays* à la table *continent* via la clé étrangère *id_continent*

12.1.2 Dans l'interface phpMyAdmin

Voici les étapes à suivre : ouvrir l'interface du Concepteur

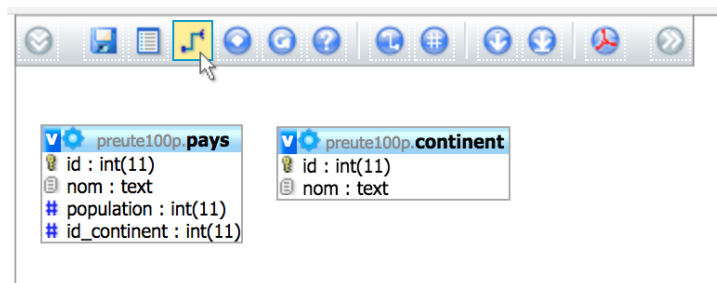


	Table	Action	Enregistrements ¹	Type	Interclassement	Taille	Perte
☐	continent	      	2	InnoDB	latin1_swedish_ci	16,0 Kio	-
☐	pays	      	3	InnoDB	latin1_swedish_ci	32,0 Kio	-

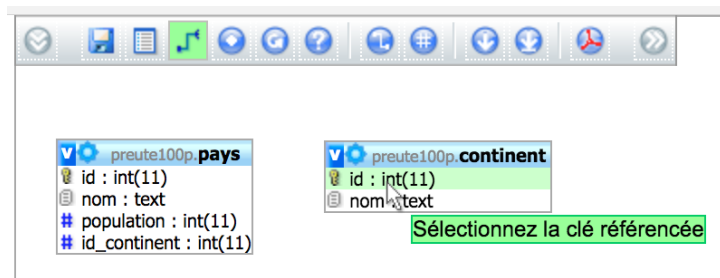
Faire dévoiler tous les champs si ce n'est pas déjà fait :



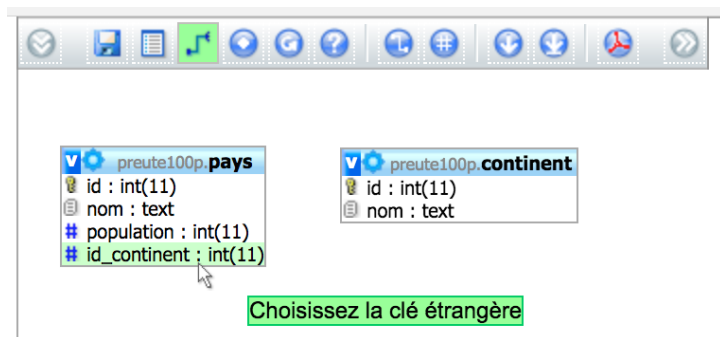
Sélectionnez l'icône pour ajouter une relation :



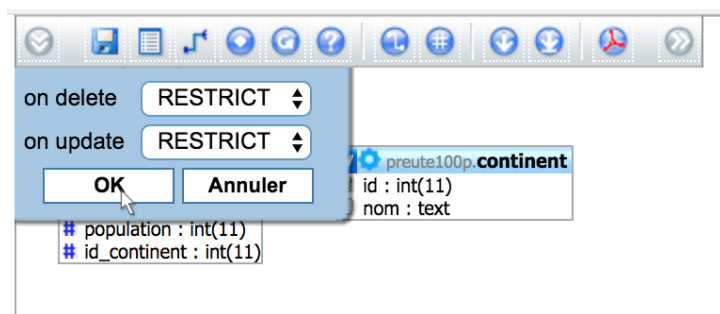
Choisissez la **clé primaire** à laquelle vous voulez faire référence



Choisissez la clé étrangère :



Choisissez le comportement lors de la suppression ou mise à jour des champs :



Sauvegardez la vue Concepteur pour obtenir les mêmes positions des tables lors d'une prochaine ouverture du concepteur :

