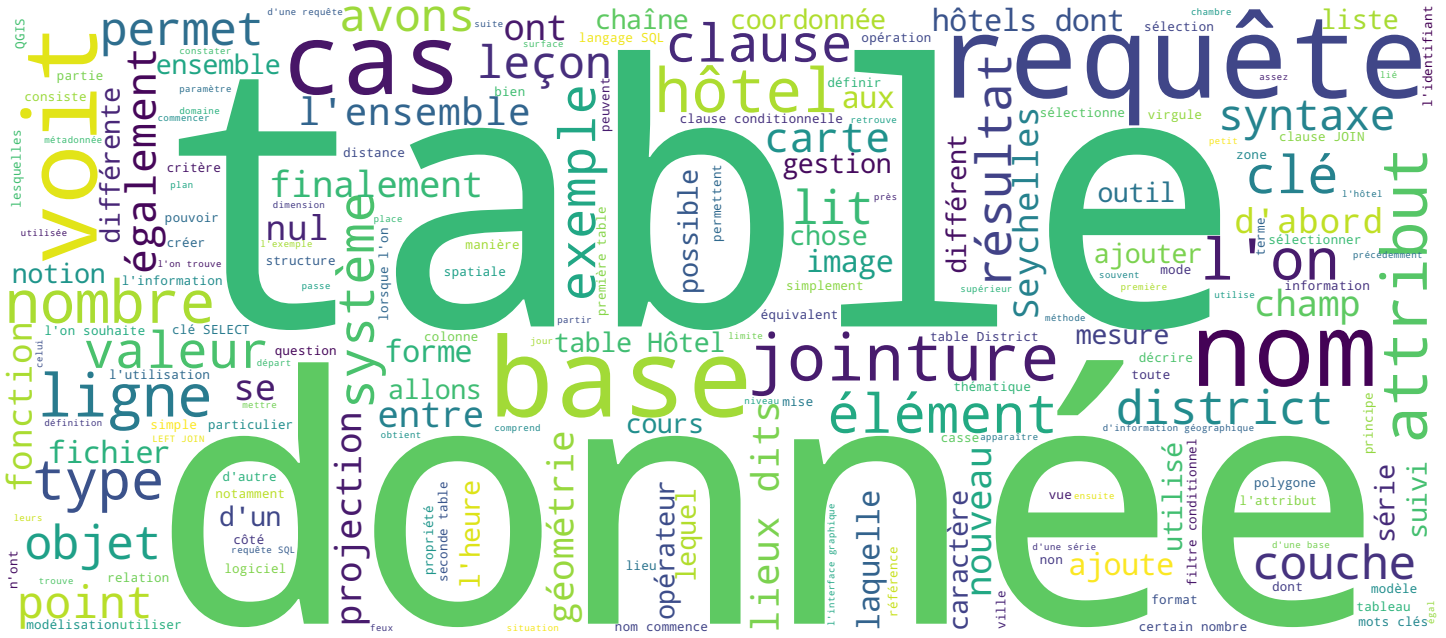
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

RESCIF
Réseau d'excellence des sciences de l'ingénieur de la Francophonie

Requêtes conditionnelles

Introduction aux systèmes d'information géographique

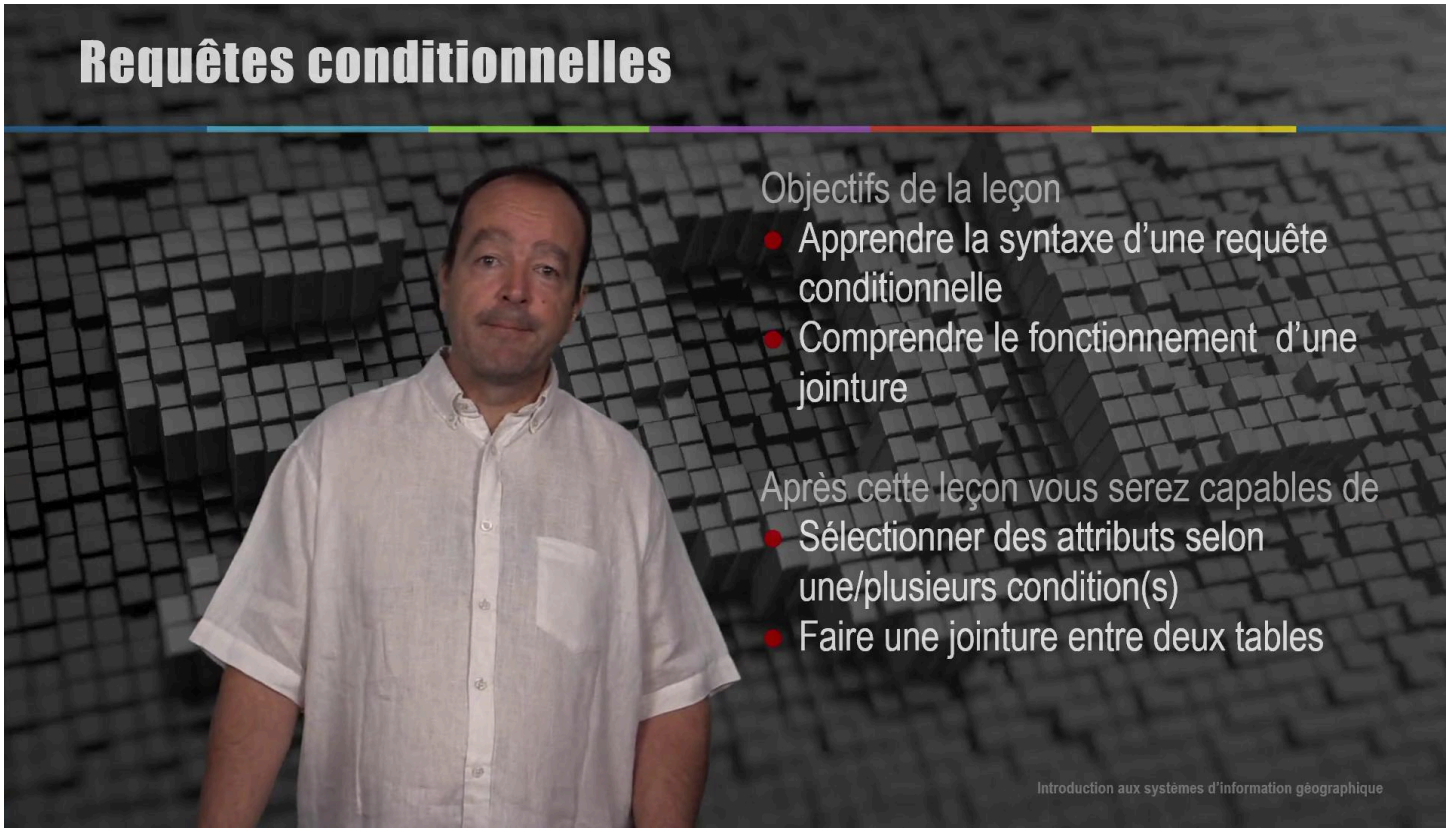
Stéphane Joost, Marc Soutter, Fernand Kouamé, Amadou Sall



Search MOOC

Video

EPFL

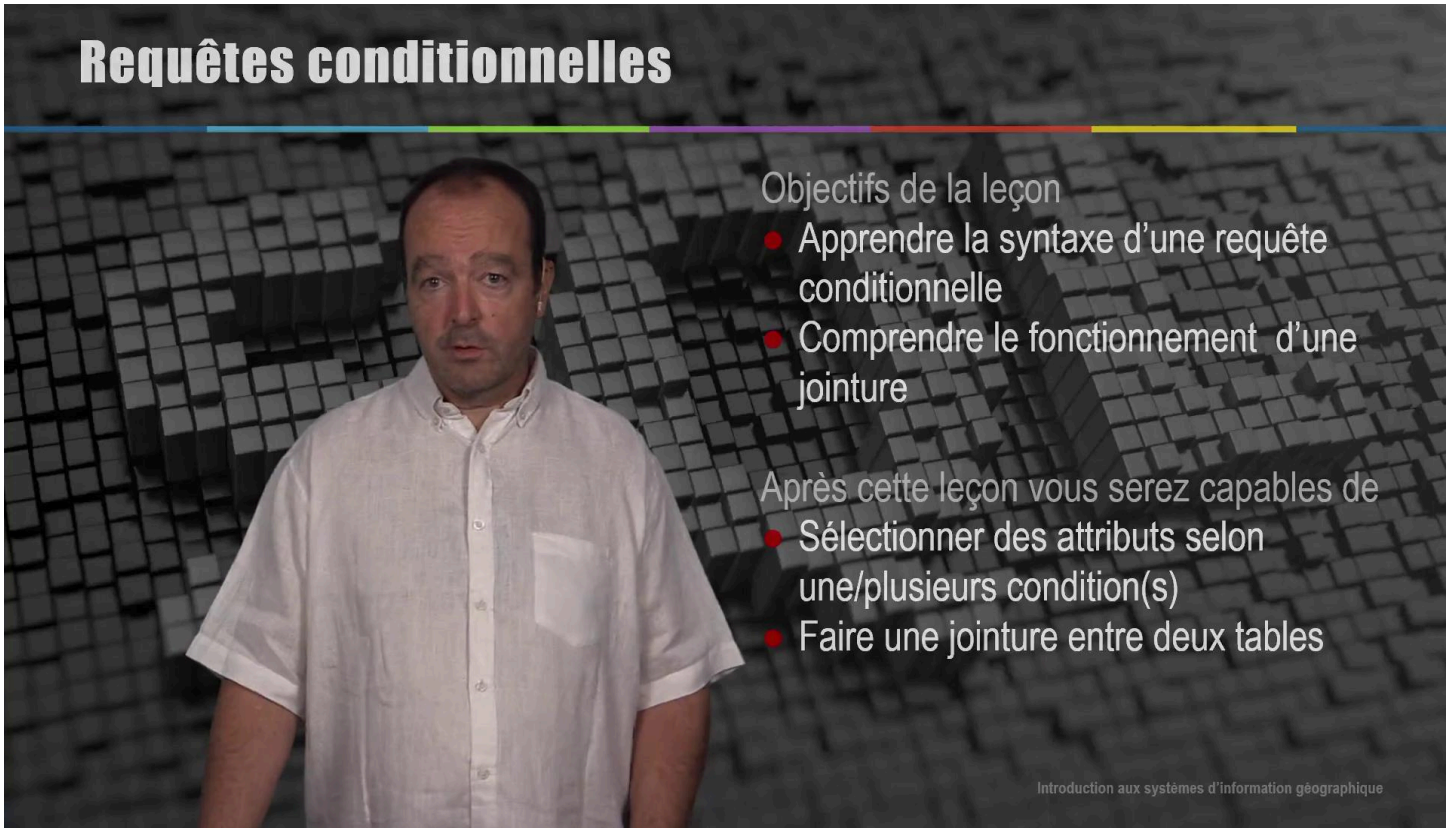


Bienvenue à cette leçon sur les requêtes conditionnelles. Ce type requête permet d'extraire un sous-ensemble d'un groupe d'objets. Par exemple, on peut extraire un ensemble de voitures rouges d'un parking sur lequel sont parqués une masse énorme de véhicules. Mais vous pouvez pas imaginer comme... Avec même des voitures mobility. Avec même des voitures mobility. Mobility. Et t'as la carte. Et t'as la carte, mobility et tu trouves plus ta voiture. Toujours quand tu dis : « J'ai besoin d'une voiture rouge. » Là, les voitures, celles-là qui n'ont pas été avec nous dès le départ, elles n'ont plus le plein [inaudible]... Oui, mais formule-le en SQL, parce que là, ils comprennent rien Nous allons donc aborder dans cette leçon la question des requêtes conditionnelles. Des requêtes qui permettent d'extraire des données sur la base d'un critère attributaire. Dans l'exemple qui nous accompagne tout au long de ce cours sur les Seychelles, des requêtes qui permettraient par exemple d'extraire le sous-ensemble des routes asphaltées. L'objectif de la leçon est donc d'étudier le principe de la syntaxe d'une requête conditionnelle et de comprendre comment ces requêtes débouchent sur la notion de jointures qui permettent d'associer plusieurs tables.

Notes

Summary

0m 21s



De sorte qu'au terme de la leçon, vous soyez en mesure d'utiliser, d'écrire des requêtes conditionnelles, de sélectionner, de filtrer des données sur la base d'un critère attributaire et d'utiliser des jointures de tables.

Notes

Summary

1m 44s

Nous verrons donc dans cette leçon, d'abord la notion de filtre conditionnel basée sur l'utilisation de la clause WHERE dans une requête SQL. Puis, les divers opérateurs que l'on peut utiliser dans ces clauses. Ensuite, les jointures qui reposent sur la clause WHERE. Et pour conclure, les jointures basées sur un autre type de clause qui est la clause JOIN. Nous avons vu dans la précédente leçon d'introduction au langage SQL que celui-ci repose sur une syntaxe de base qui comprend un certain nombre de mots-clés, définitions des clauses, des clauses de sélection, de filtres conditionnels d'agrégations, et cetera. Dans la présente leçon, nous abordons la clause de filtre conditionnel qui s'exprime par le mot-clé WHERE assorti d'une condition. La syntaxe de base de cette requête comprend tout d'abord le mot SELECT suivi du nom des attributs ou du métacaractère astérisque lorsque l'on veut sélectionner tous les attributs, du mot clé FROM suivi du nom de la table de laquelle on va tirer les informations et finalement le mot-clé WHERE suivi de la condition. Condition qui est composée par trois éléments : un attribut, un opérateur et un critère.

Notes

Summary

1m 57s

Filtre conditionnel - clause WHERE

```
SELECT * FROM nom_table WHERE condition
```

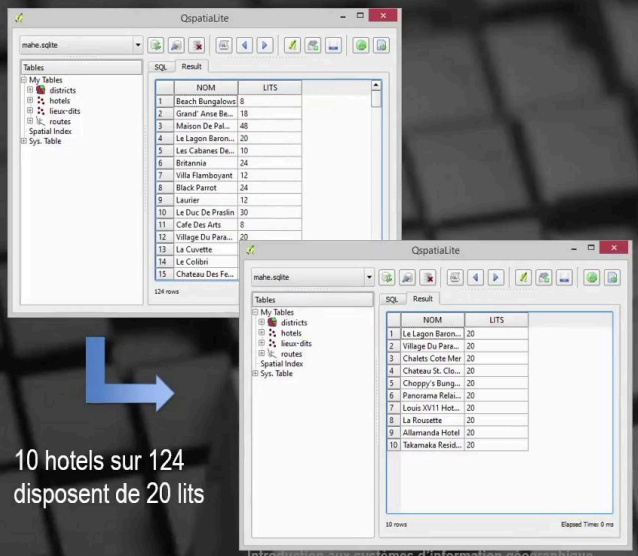
Condition composée par

- Un attribut
- Un opérateur
- Un critère

Exemple: recherche des hôtels de 20 lits

```
... WHERE hotels.lits = 20
```

Attribut Opérateur Critère



10 hotels sur 124
disposent de 20 lits

Si l'on prend l'exemple de la recherche des hôtels de 20 lits aux Seychelles, on voit que la condition qui se trouve dans la clause conditionnelle comprend comme attribut le mot-clé « hotels.lits ». Donc, l'attribut « lits ». Comme opérateur, le signe égal et comme critère la valeur 20. Dans le cas de la base de données des Seychelles, nous avons donc un ensemble de 124 hôtels. Et si l'on applique cette requête de filtre conditionnel sur le critère Nombre de lits = 20, on voit que l'on extrait une série de 10 hôtels qui compte exactement 20 lits.

Notes

Summary

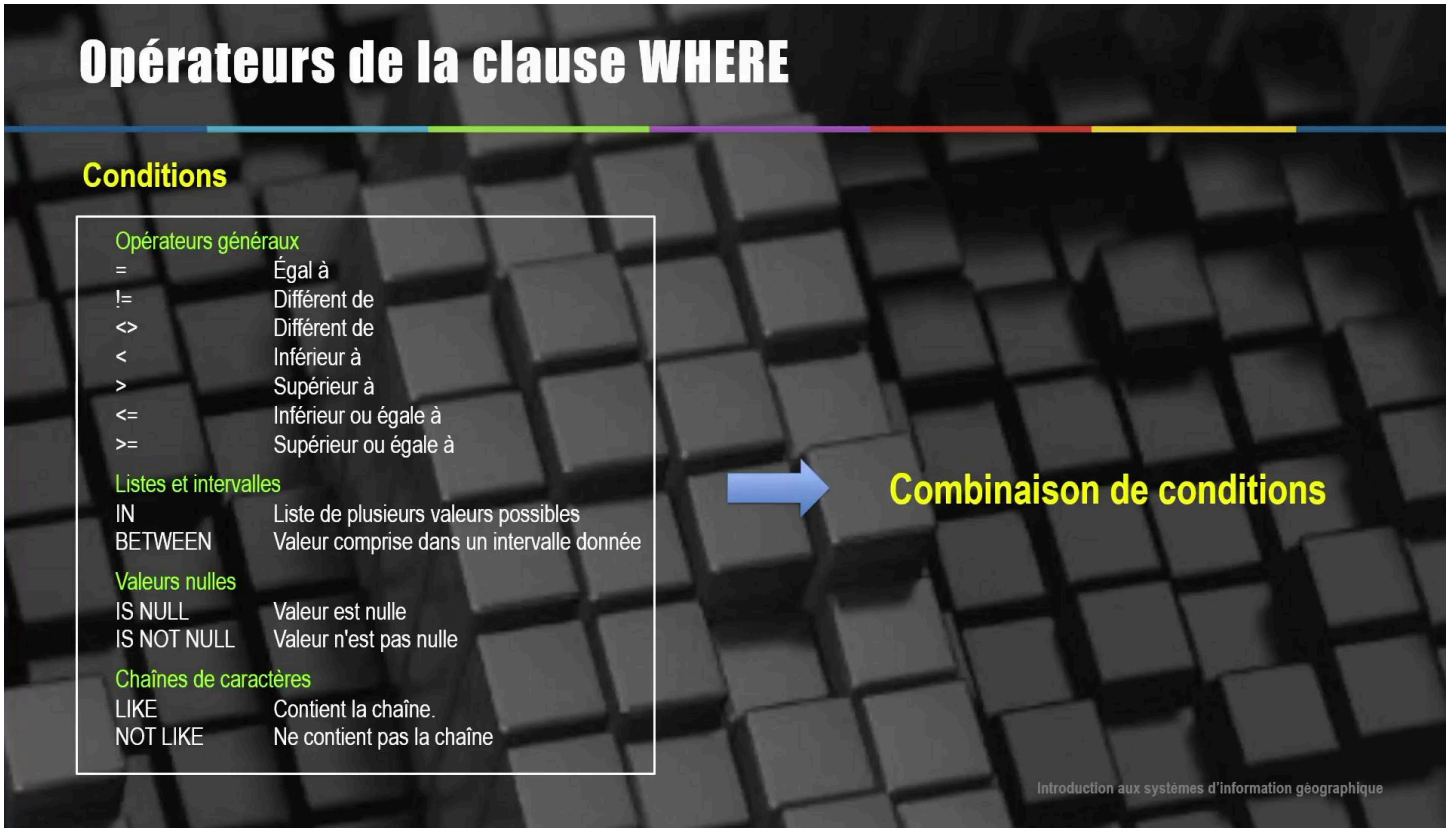
3m 16s

Les opérateurs que l'on peut utiliser dans une clause WHERE sont de différentes natures, à commencer par les opérateurs généraux. Et en premier lieu, l'opérateur « égal » qui permet de comparer deux valeurs entre elles et son alter ego, l'attribut « différent de » qui peut s'exprimer soit sous la forme d'un point d'exclamation et du signe égal, soit de deux signes < et >. Puis les deux attributs inférieur et supérieur à, ou inférieur ou égal à. Les intervalles, les listes avec pour les listes, le mot-clé IN suivi d'une série de valeurs séparées par des virgules et enfermées dans des parenthèses qui expriment l'idée que la valeur de l`attribut est présente dans la collection, l'échantillon de valeur proposée. Pour les intervalles, le mot-clé BETWEEN suivi de deux valeurs séparées par le mot-clé AND qui exprime l'idée que la valeur de l'attribut est comprise entre les deux bornes définies par valeur 1 et valeur 2. Le test sur les valeurs nulles avec les mots-clés IS NULL, IS NOT NULL pour vérifier si la valeur de l`attribut est nulle ou n'est pas nulle. Finalement, les opérateurs d'inclusion et d'exclusion de chaînes de caractères qui permettent de travailler sur des chaînes de caractères avec tout d'abord l'opérateur d'égalités qui permet de comparer une chaîne de caractère à une valeur entre crochets.

Notes

Summary

4m 03s



Un équivalent du signe =, c'est l'opérateur LIKE. À ceci près qu'il est non sensible à la casse, donc, indépendant des majuscules, minuscules utilisées pour décrire la chaîne de caractères Valeurs. Ceci est vrai dans le cas de SQLite, mais elle n'est pas systématique avec tous les systèmes de base de données. L'opérateur LIKE avec les métacaractères, ici, dans le langage SQL, c`est le signe % qui est utilisé comme metacaractère, donc comme caractère de remplacement pour des chaînes de caractères. Donc, on va chercher l'ensemble des objets dont l`attribut comprend la chaîne de caractère Valeurs à un endroit ou un autre. Et puis finalement l'opérateur NOT LIKE pour dire qu'une valeur n'est pas comprise. Comme on le dit dans PostgreSQL, LIKE et = sont équivalents et on a un autre mot-clé, un autre opérateur, ILIKE, qui est, lui, insensible à la casse. Ce tableau fait la synthèse des différents types d'opérateurs que nous venons de voir, donc, les opérateurs généraux, les listes et intervalles, les valeurs nulles et les opérateurs portant sur les chaînes de caractères. Ces opérateurs sont utilisés pour définir des conditions, et ce qui est intéressant dans les clauses WHERE, c'est que ces conditions peuvent-être combinées pour effectuer des recherches élaborées.

Notes

Summary

5m 33s

La combinaison de conditions repose sur les mots-clés AND et OR qui permettent d'associer de manière inclusive ou exclusive, deux conditions. Les conditions supplémentaires peuvent être ajoutées à ce système en utilisant les règles usuelles de parenthèses.

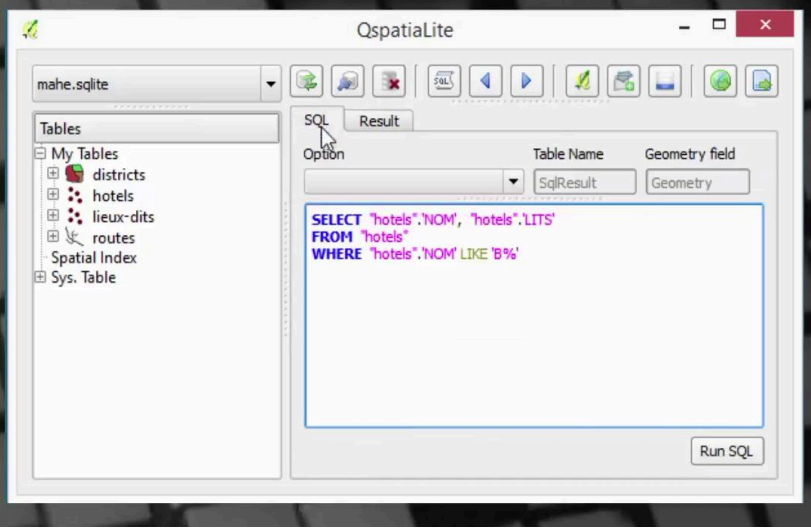
Notes

Summary

7m 03s

Opérateurs de la clause WHERE

Combinaison de conditions



Nous illustrons ici ces opérateurs dans le cas d'une base de données QspatiaLite en sélectionnant dans la table Hôtels, les noms et les lits et en ajoutant comme clause conditionnelle, le fait que les lits doivent être égal à 20. On effectue cette requête. En modifiant, cette requête, on peux rechercher l'ensemble des hôtels qui ont moins de 20 lits, l'ensemble des hôtels qui ont plus de 20 lits évidemment, l'ensemble des hôtels qui ont 158 ou 176 lits. Et on en trouve deux qui ont exactement 158 et 176 lits. Alternativement, on peut rechercher l'ensemble des hôtels dont le nombre de lits est compris entre ces deux valeurs de 158 et de 176. On constate qu'on en trouve trois. Ce qui permet aussi de constater que les limites, les bornes 158 et 176 sont inclusives et non exclusives. On recherche ensuite ici l'ensemble des hôtels dont le nom correspond à Banyan Tree. On trouve cet hôtel et on voit que si on avait écrit Banyan Tree avec un T minuscule, la requête n'aurait donné aucun résultat alors que dans le cas de QspatiaLite avec un LIKE, la requête n'est pas sensible à la casse, et on trouve l'hôtel recherché. On recherche ensuite l'ensemble des hôtels dont le nom commence par B, et l`on en trouve 13.

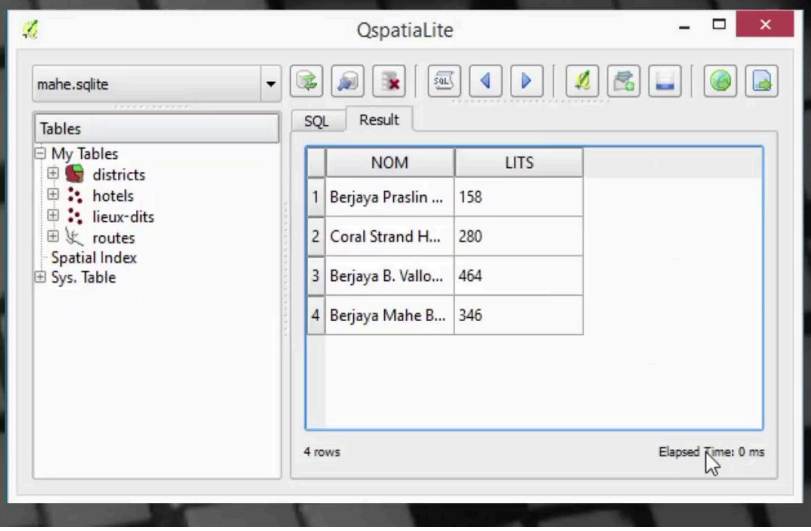
Notes

Summary

7m 25s

Opérateurs de la clause WHERE

Combinaison de conditions



Puis, l'ensemble des hôtels dont le nom commence par B ou dont le nom commence par C. On ajoute une condition supplémentaire : le fait que le nombre de lits doit être supérieur à 100. On voit que l'on trouve quatre candidats, qui vont répondre à ces critères.

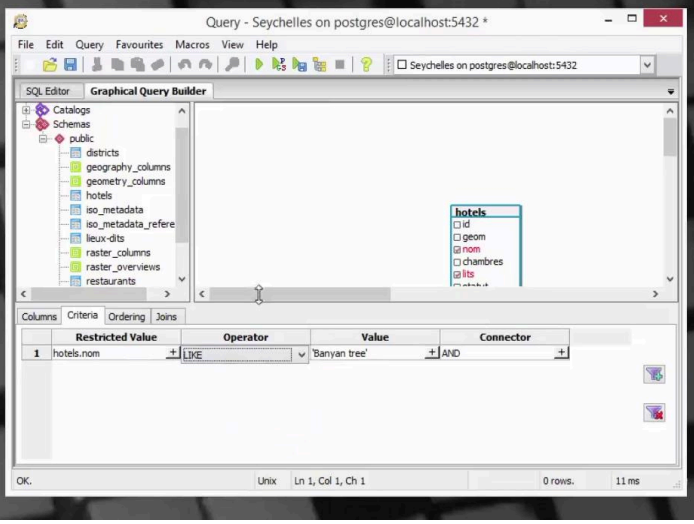
Notes

Summary

9m 16s

Opérateurs de la clause WHERE

Combinaison de conditions



Nous effectuons maintenant la même série d'opérations dans le cas d'une base de données PostGIS en utilisant l'interface pgAdmin. À nouveau, on sélectionne, dans la table des hôtels, les champs Noms et Lits. On ajoute un critère dans l'interface graphique pour que le nombre de lits égal à 20, On change l'opérateur dans cette requête pour avoir le nombre des hôtels dont le nombre de lits est inférieur à 20, maintenant supérieur à 20. Comme précédemment, on va rechercher les hôtels dont le nombre de lits vaut 158 ou 176 et on voit que l'on doit écrire dans cette interface, l'ensemble de la condition, de la même manière que dans le cas de QspaciaLite où on fait ça en pur SQL. Donc ici, qu`on utilise dans ce cas-là, l'interface graphique ou non, ne fait pas de différence. La requête pour les hôtels dont le nombre de chambres est entre 158 et 176, vous l'avez vu passer. Et maintenant, on recherche l'hôtel qui s'appelle Banyan Tree que l'on trouve bien évidemment. Le cas de figure avec un T minuscule et on voit qu'effectivement on n'a pas de résultat. En remplaçant l'opérateur = par l'opérateur LIKE, toujours pas de résultat.

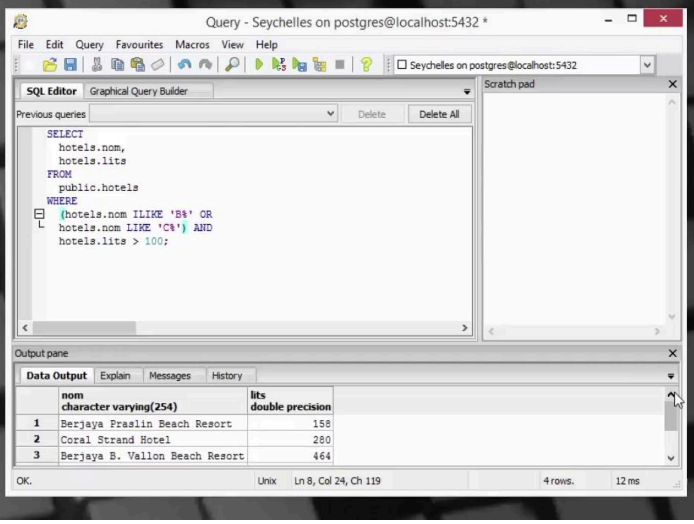
Notes

Summary

9m 39s

Opérateurs de la clause WHERE

Combinaison de conditions

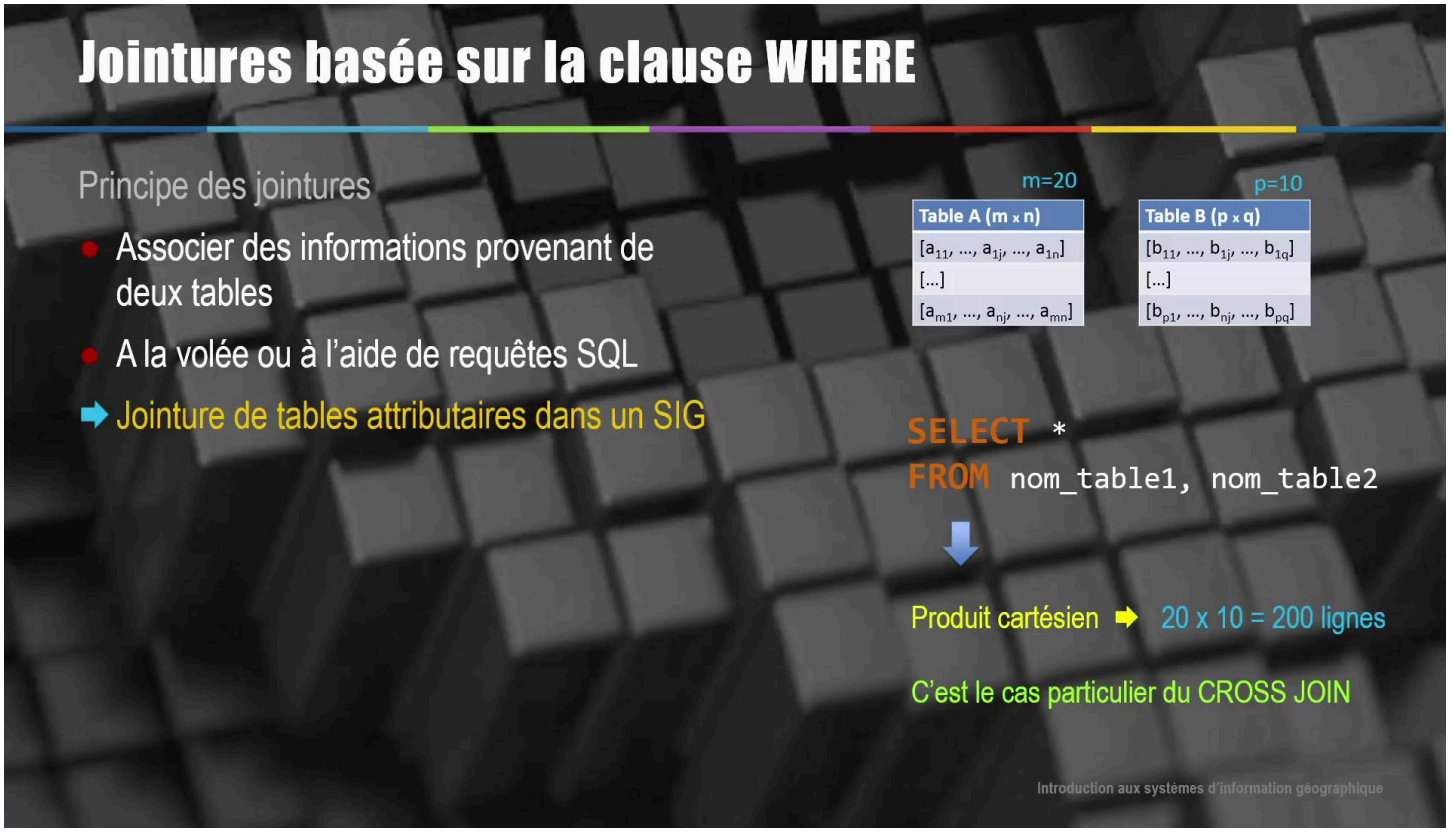


Comme je vous l'ai dit, LIKE n'est pas non plus insensible à la casse, on doit utiliser l'opérateur ILIKE pour pouvoir faire une requête qui ne soit pas sensible à la casse. Comme précédemment, on recherche les hôtels dont le nom commence par B, on en trouve également 13, ce qui est rassurant. On associe maintenant une requête supplémentaire avec de nouveau le nom de l'hôtel qui commence cette fois par la lettre C. Et puis, troisième requête complémentaire, le nombre de lits supérieur à 100, comme tout à l'heure. Comme tout à l'heure, on trouve quatre candidats. Il s'est passé quelque chose de curieux. Oui. Les parenthèses au bon endroit. On trouve nos quatre candidats.

Notes

Summary

11m 20s

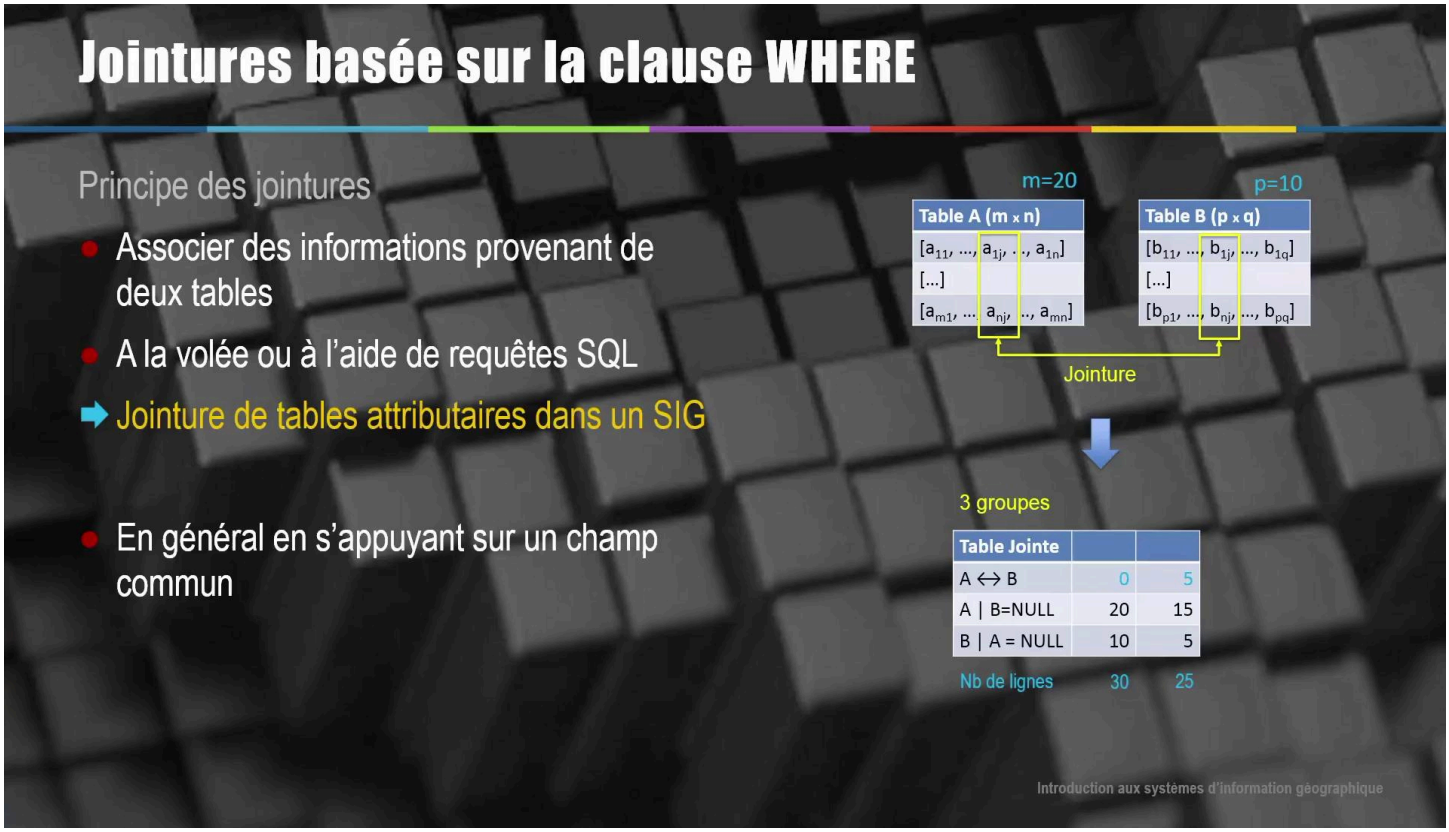


Le principe des jointures consiste donc à associer des informations provenant de deux ou davantage de tables et peut être appliquer soit à la volée soit à l'aide de requêtes SQL. Nous avions vu dans la leçon portant sur la modélisation des données, un exemple de jointure de table attributaire à la volée effectuée dans le logiciel QJis. Ici, on va s'intéresser à à la manière d'utiliser le langage SQL pour effectuer ces jointures de tables. Supposons que l'on ait deux tables, A et B, constituées d'une série de lignes qui contiennent des objets qui ont eux-mêmes un certain nombre d'attributs de A1 à N. Et pareillement pour la table B qui a des dimensions P par Q. Supposons que la table A compte 20 lignes et la table B, 10 lignes. La requête d'association de ces tables standard serait du type SELECT [inaudible 00:13:38] du nom des attributs, le mot-clé FROM, et puis les deux noms de tables qui se suivent, séparés par une virgule. Ce type de requête donnerait le produit cartésien de ces deux tables, donc chaque élément, chaque ligne de la table A serait multiplié par chaque ligne de la table B ou associé à chaque ligne de la table B, de sorte à produire un ensemble résultant de 200 lignes, Cet ensemble résultant de 200 lignes correspond au cas particulier de jointure qu'on appelle le CROSS JOIN.

Notes

Summary

12m 36s

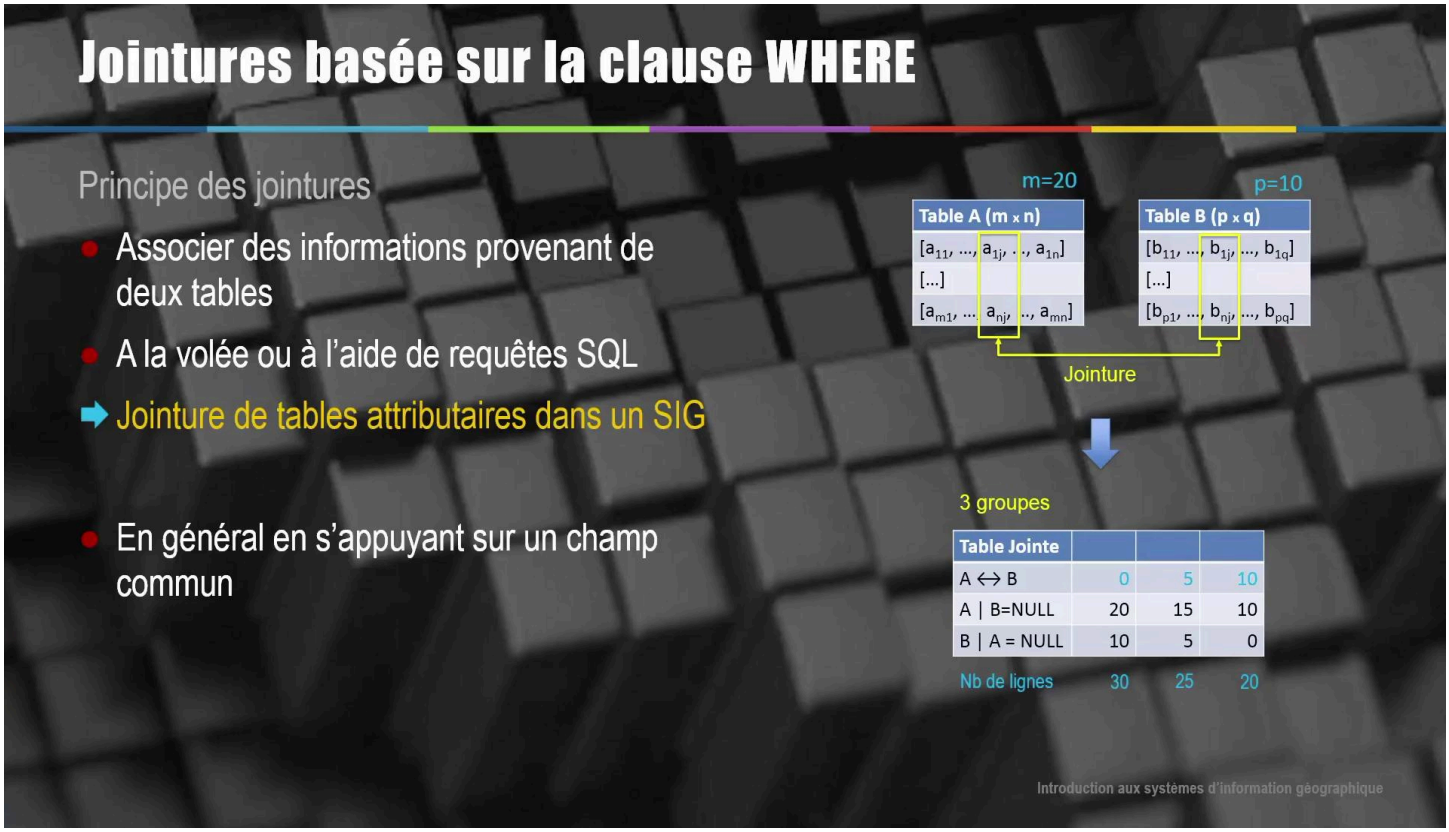


La jointure de table s'appuie toutefois, le plus souvent sur un champ commun qui permet d'associer ensemble les lignes des deux tables pour lesquelles ce champ commun a la même valeur. Ce type de jointure conduit à distinguer trois types de situations dans les résultats que l'on produit, dans les comparaisons entre lignes des deux tables. Tout d'abord, le cas où les deux lignes sont jointes par un champ commun, qui est la même valeur. Après, on a les situations où les lignes de la table A qui n'ont pas d'équivalent de lignes jointes dans la table B, donc où B est nul. Inversement, les lignes de la table B qui n'ont pas d'équivalent du côté de A donc pour lesquelles A est nul. Si l'on regarde la taille des tables de résultats obtenus, on voit que si le nombre de jointures est nul, donc, il existe aucune ligne des deux tables qui ont une valeur commune pour le champ joint, on trouve un ensemble de 30 lignes dans le résultat, donc les 20 lignes de la table A pour lesquelles B est nul et les 10 lignes de la table B pour lesquelles A est nul. Dans le cas où on aurait cinq correspondances, cinq lignes de chacune des tables jointes entre elles, on voit que les lignes de A, qui auraient une correspondance nulle, sont au nombre de 15.

Notes

Summary

14m 12s



Inversement, les lignes de B, qui seraient nulles du côté A, sont au nombre de cinq. On aurait comme résultat final, un total de 25 lignes dans une requête qui reprendrait l'ensemble de ces valeurs. Et finalement dans le même esprit, si on a dix jointures, on retrouve au total 20 résultats possibles.

Notes

Summary

15m 44s

Jointures basée sur la clause WHERE

```
SELECT nom_table1.nom_attribut1, nom_table2.nom_attribut2
FROM nom_table1, nom_table2
WHERE nom_table1.nom_attribut3 = nom_table2.nom_attribut4
```

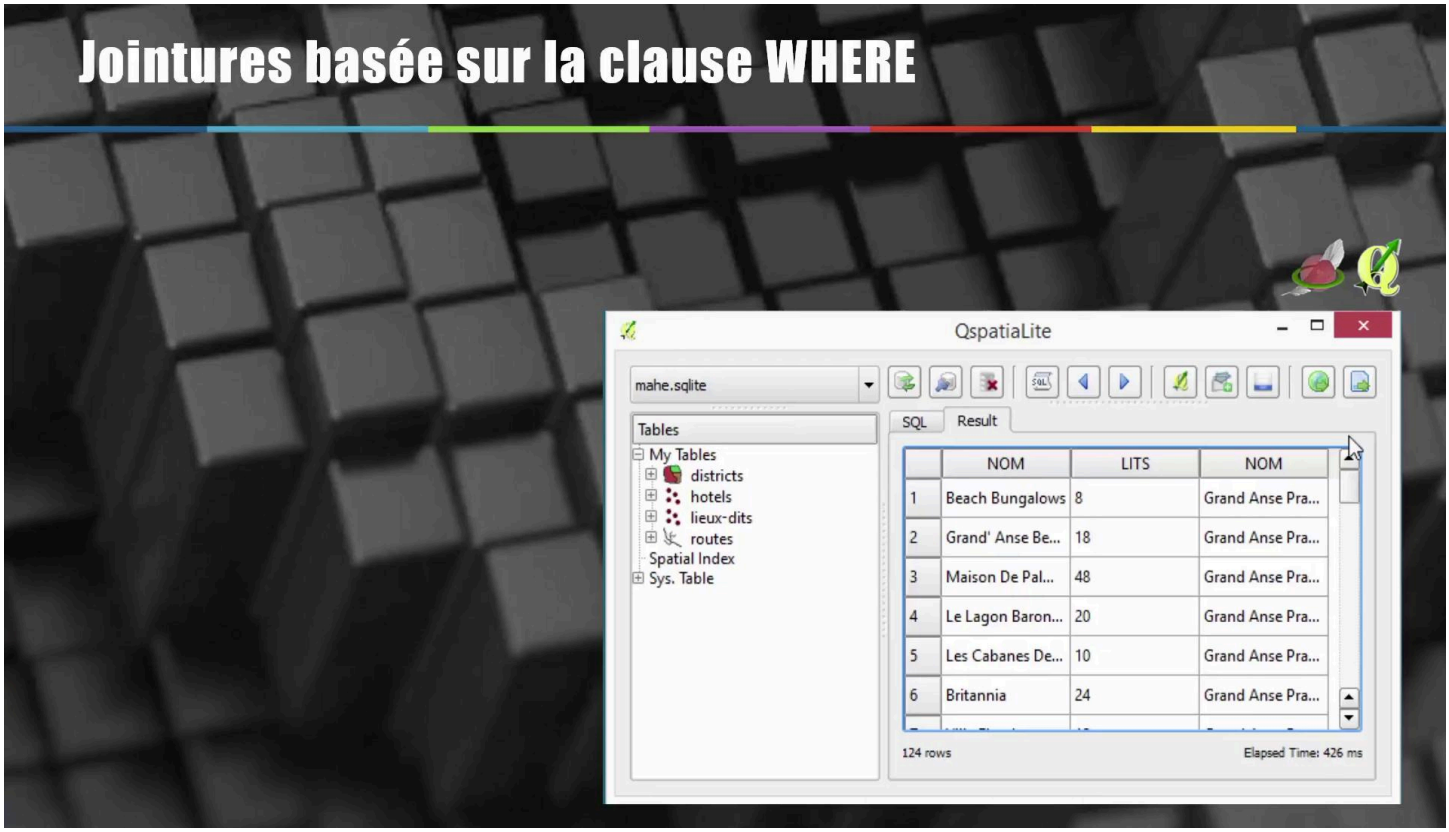

La syntaxe générale, donc une jointure basée sur la clause WHERE, ressemble à ceci. Tout d'abord, le mot-clé SELECT, suivi de l`attribut de la première table et de l`attribut de la seconde table, séparé par une virgule, le mot-clé FROM suivi des deux noms de table, séparé par une virgule à nouveau et puis la clause WHERE avec la condition qui associe un attribut de la première table, l'opérateur d'égalité et comme critère, un attribut de la seconde table, les deux attributs de la clause WHERE définissant la jointure.

Notes

Summary

16m 08s

Jointures basée sur la clause WHERE



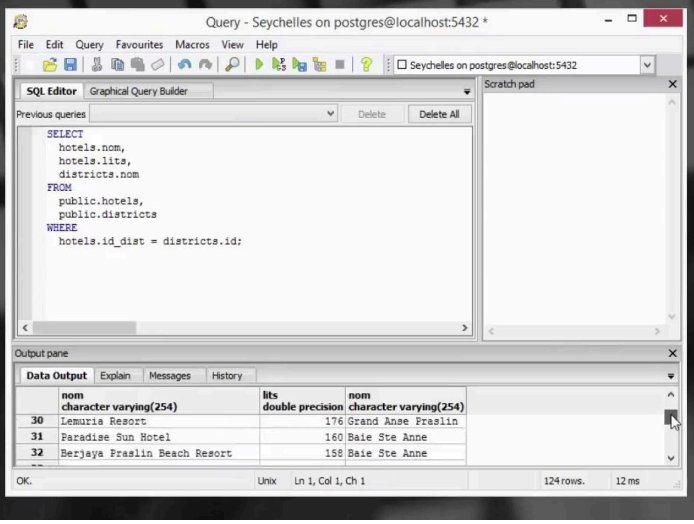
Reprenons le cas de la base de données QspatiaLite sur les Seychelles. On écrit un peu la même requête que tout à l'heure pour les lits des hôtels. Donc le mot-clé SELECT, les champs Nom et Lits, nombre de lits de la table hôtel, le mot-clé FROM avec la table hôtel et puis la clause conditionnelle dans laquelle, on exprime l'idée que l'identifiant de districts qui est associé à l'hôtel est égal à son équivalent dans la table Districts. On ajoute le nom de districts au résultat recherché et puis le nom de la table Districts. Et on obtient la liste des 124 hôtels des Seychelles avec leur nombre de lits et le nom du district dans lequel ils se trouvent.

Notes

Summary

16m 51s

Jointures basée sur la clause WHERE

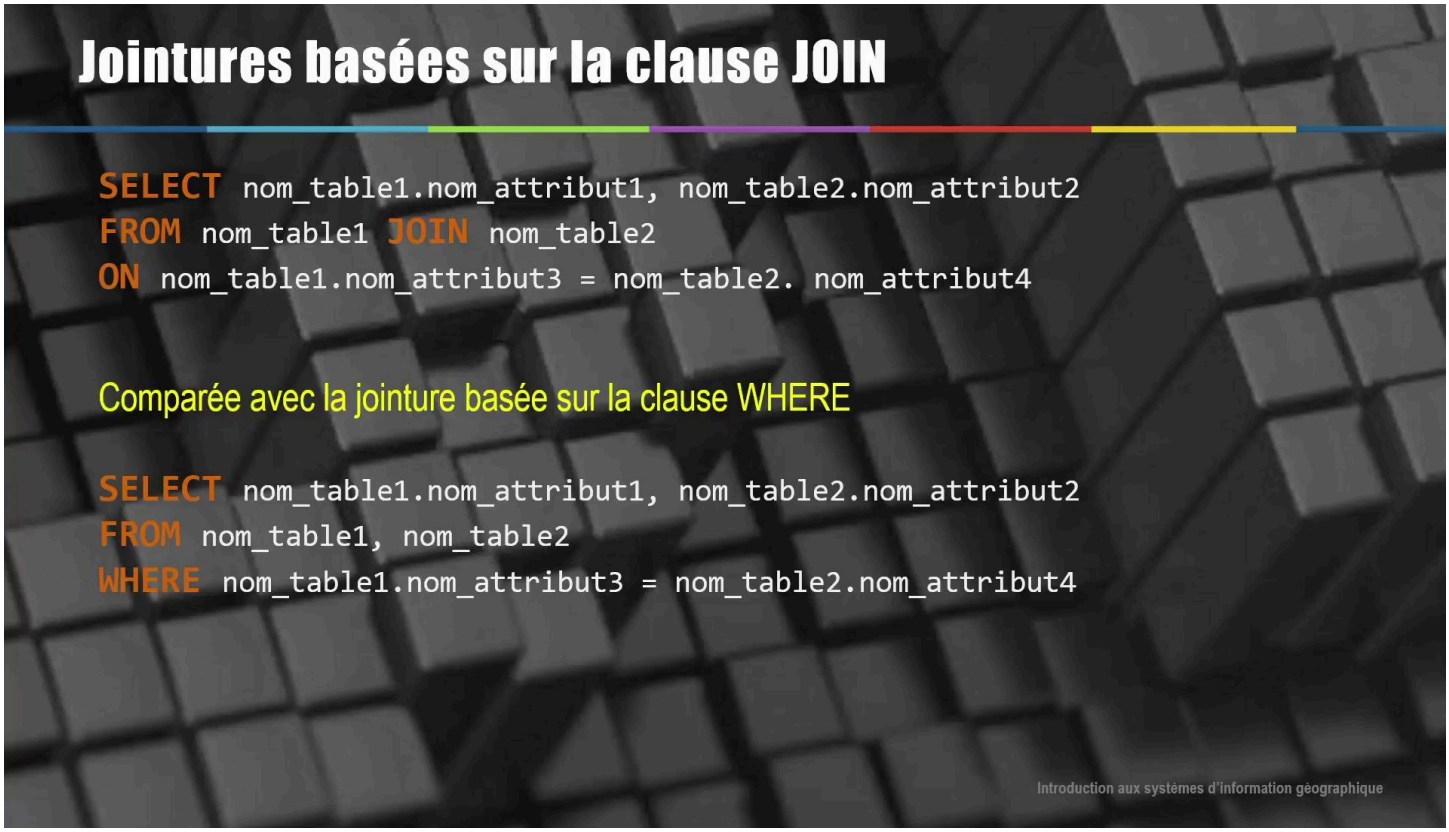


La même opération maintenant, dans l'interface pgAdmin de la table Postgres/PostGIS où on lie graphiquement les deux tables. Par le champ identifiant, on sélectionne les champs que l'on souhaite voir apparaître dans le résultat et on exécute la requête directement.

Notes

Summary

17m 40s



La syntaxe de base d'une requête de jointure basée sur la clause JOIN se présente de la manière suivante. Tout d'abord, le mot-clé SELECT, les attributs concernés que l'on souhaite voir apparaître dans le résultat, donc, un attribut de la table 1, un attribut de la table 2, la jointure qui va de la table 1 vers la table 2 et puis le mot-clé ON pour spécifier le champ sur lequel s'effectue cette jointure. Dans le cas présent, l`attribut 3 de la table 1, qui est mis en correspondance avec l`attribut 4 de la table 2. Si l`on compare cette syntaxe avec celle que nous venons de voir dans le cas de la jointure basée sur la clause WHERE, on voit que la différence est très mince, puisque l'on introduit simplement le mot JOIN pour séparer les tables, et puis le mot-clé ON à la place de la clause WHERE pour définir le critère de la jointure. L'intérêt d'utiliser la syntaxe qui utilise la clause JOIN, c`est de bien séparer dans une requête SQL complexe les éléments de jointure des éléments conditionnels, ce qui rend la requête plus lisible.

Notes

Summary

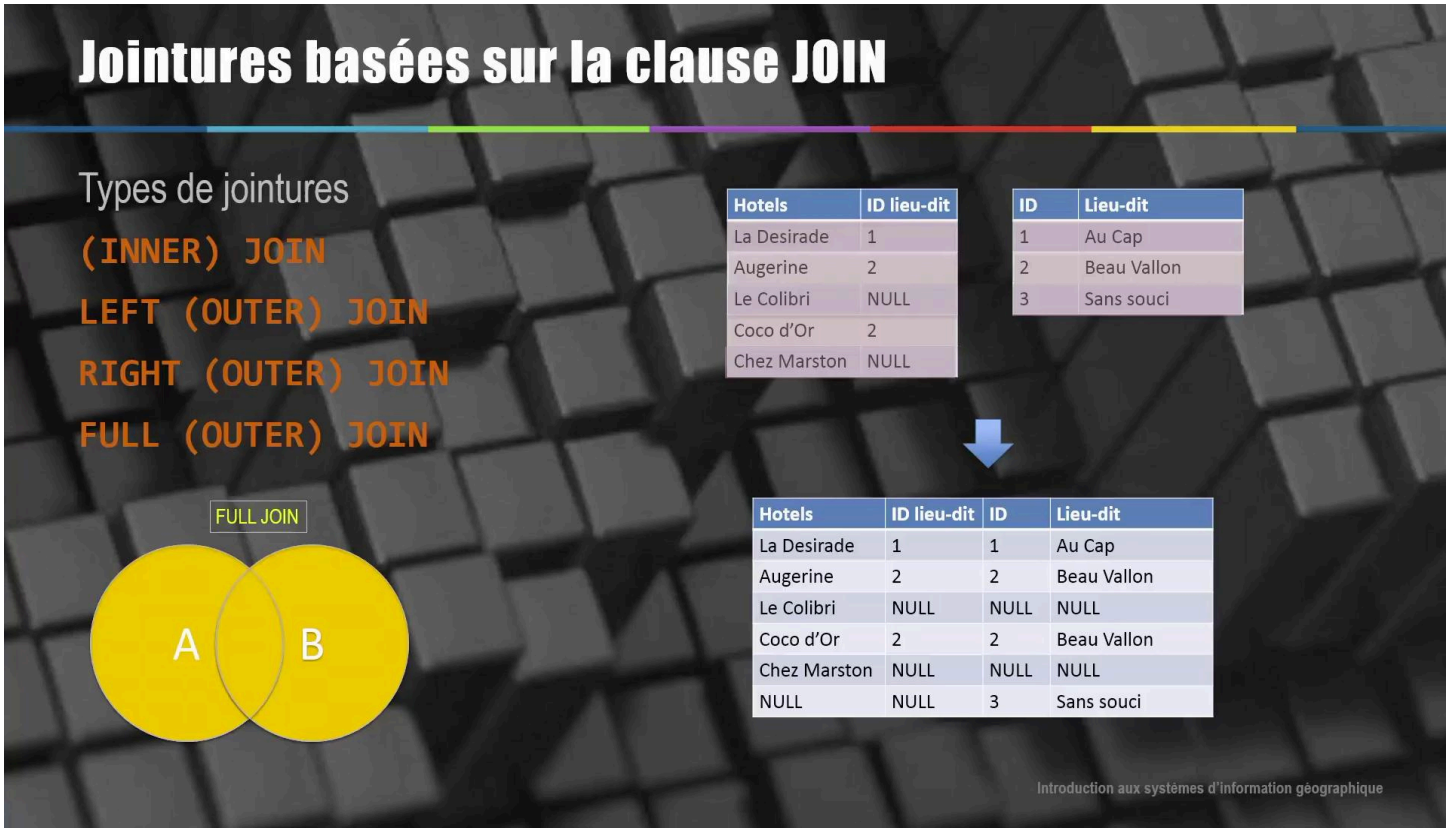
18m 16s

Il existe plusieurs types de jointures, à commencer par la jointure simple. (INNER) JOIN, le mot-clé INNER n`étant pas nécessaire. (INNER) JOIN et JOIN sont des choses équivalentes. On prend l'exemple d'une série d'hôtels associés à un lieu-dit et d'une seconde table dans laquelle on a une série de lieux-dits. On voit que dans notre exemple, les éléments qui vont être sélectionnés pour lesquels les champs ID, Lieux-dits et ID correspondent, sont au nombre de trois. Les trois hôtels La désirade, Augerine et Coco d'or qui se retrouvent dans la table de résultats. Deuxième type de jointure, le LEFT JOIN ou LEFT (OUTER) JOIN avec le mot OUTER qui est de nouveau facultatif, qui consiste à prendre l'ensemble des éléments du tableau A, de l'ensemble A auxquels on joint les éléments correspondants du tableau B. Dans notre cas, l'ensemble des hôtels de la table hôtels avec, lorsque c'est possible, les éléments d'information jointifs de la table Lieu-dit. On voit que notre résultat contient cette fois cinq éléments, avec dans deux cas des valeurs nulles pour la jointure et les paramètres joints, donc le lieu-dit.

Notes

Summary

19m 32s



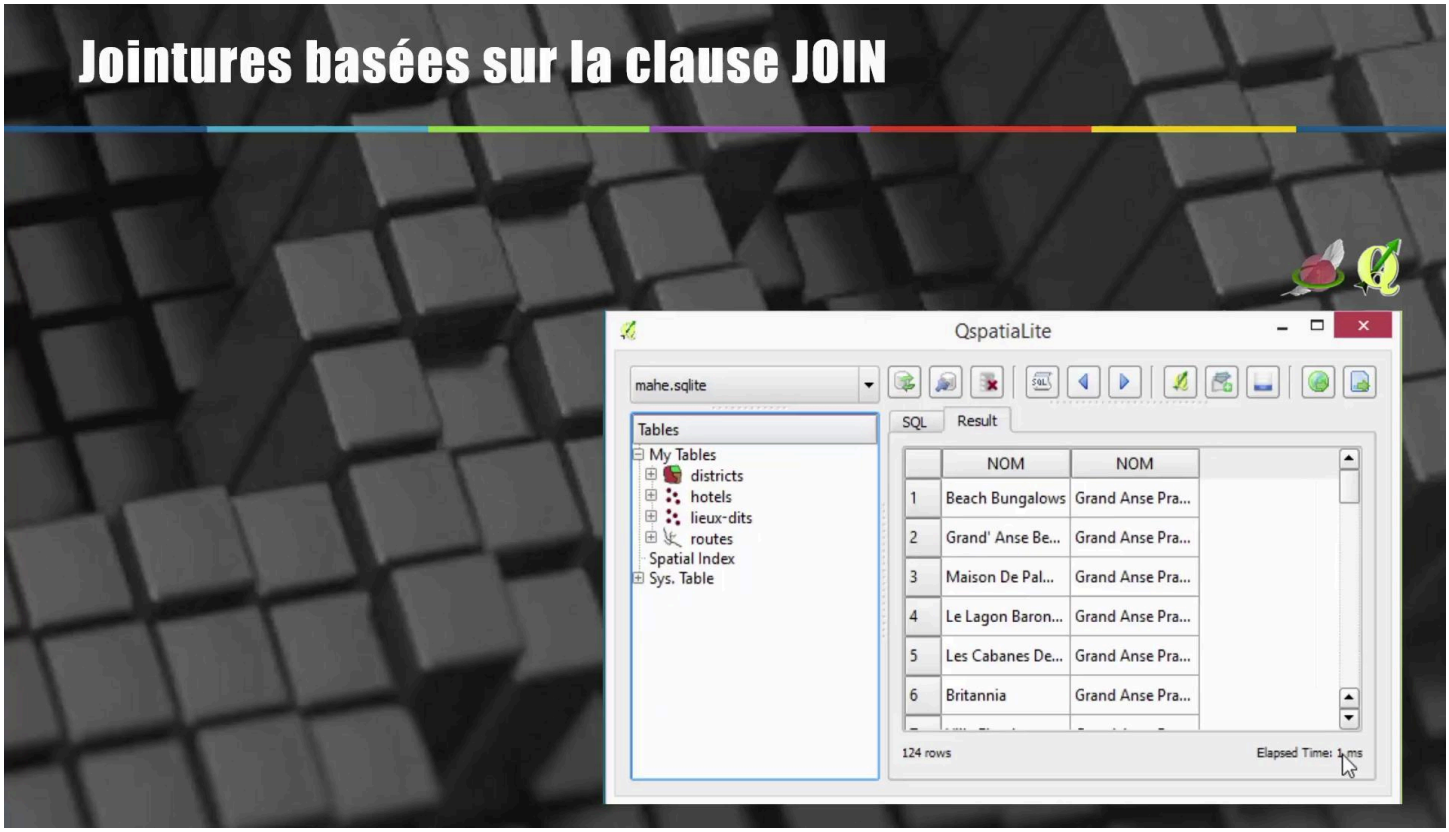
Troisième type de jointure, le RIGHT JOIN, qui permet de sélectionner l'ensemble des éléments de la seconde table donc l'ensemble des lieux-dits auxquels on associe, lorsque c'est possible, les éléments de la première table, qui ne seraient pas nuls au niveau de la jointure. On obtient comme résultat un tableau de quatre valeurs, dont les trois cas de base où la jointure existe, et puis le dernier cas, le lieu-dit Sans souci, qui n'a pas de jointure et pour lesquels les éléments de la table Hôtels sont nuls. Finalement le FULL JOIN qui consiste à prendre l'ensemble des possibles, donc la série des trois jointures où on a des objets liés de part et d'autre, plus les trois cas où soit du côté A, soit du côté B les éléments jointifs sont nuls. Et l'on obtient une table qui compte six éléments, donc deux de moins que la table de huit qu'on aurait obtenue si aucun élément jointif n'avait été présent.

Notes

Summary

21m 02s

Jointures basées sur la clause JOIN



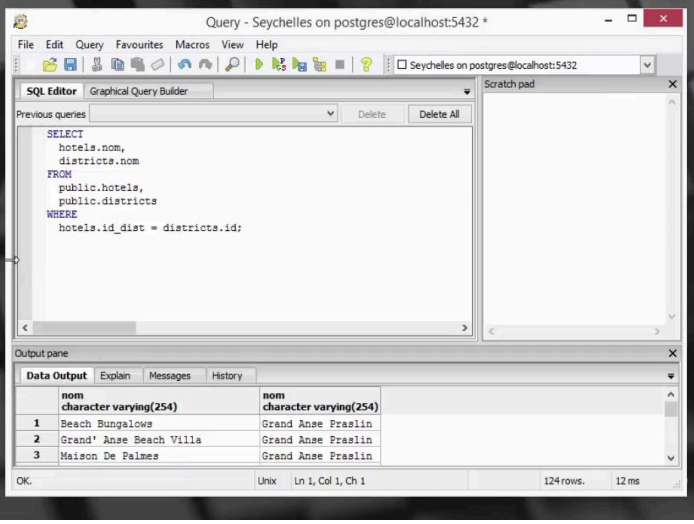
La syntaxe de ces requêtes spécifiques de jointure reste toujours la même avec simplement la clause LEFT, RIGHT ou FULL, qui est ajoutée à la clause JOIN dans la définition de la jointure. Un exemple avec la base de données QspatiaLite des Seychelles, où l'on définit dans les objets de la requête, le nom des hôtels et puis le nom des districts dans lesquels se trouvent ces hôtels, puisqu'ils sont liés par une jointure. On définie cette jointure entre la table Hôtels et la table Districts sur la base du champ Identifiant du district de la table Hôtel est équivalent à l'identifiant du district dans la table Districts.

Notes

Summary

22m 16s

Jointures basées sur la clause JOIN

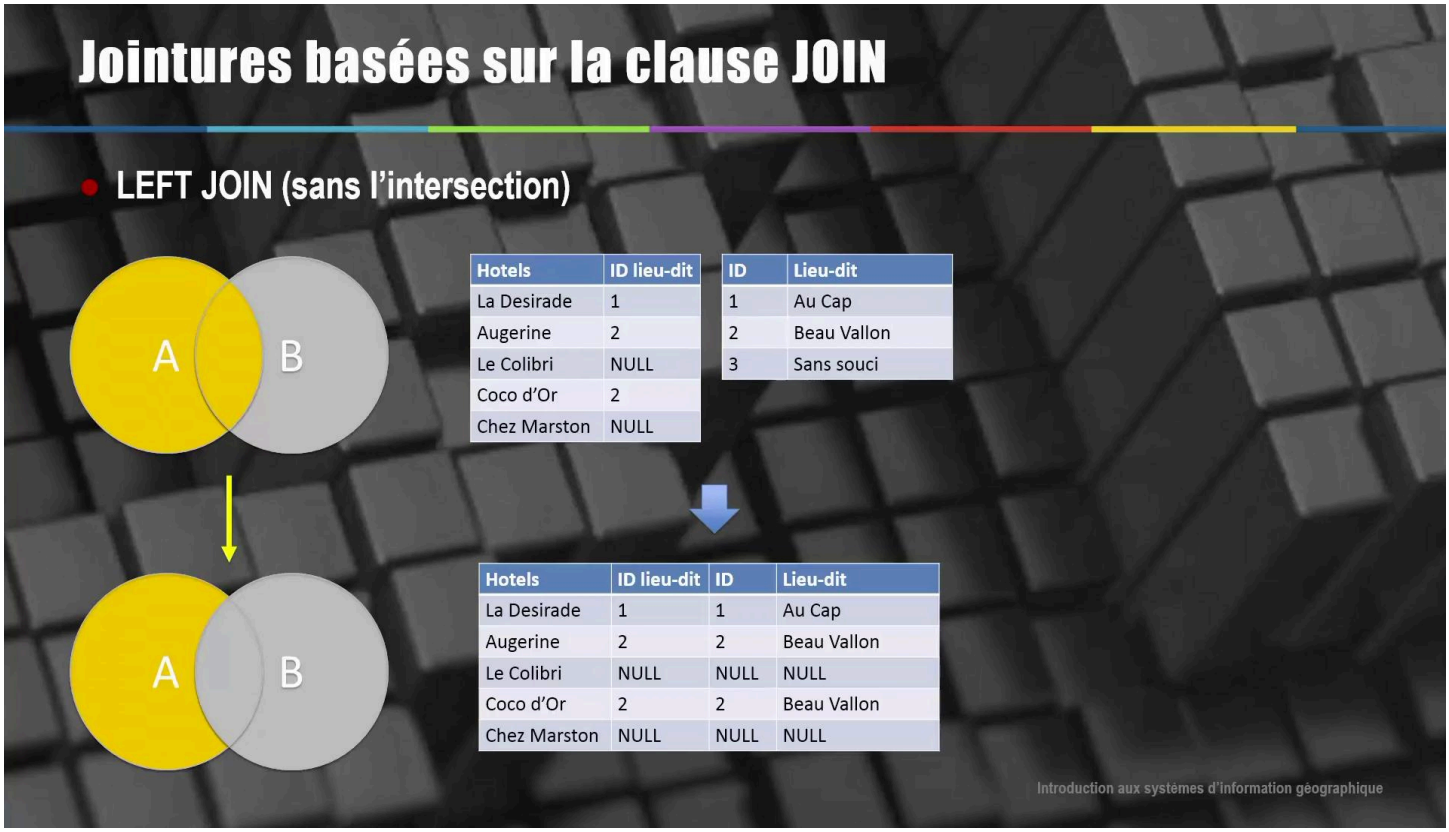


Même opération dans le cas de Podgres/PodGIS avec pgAdmin, où on ajoute graphiquement les deux tables dans le constructeur de requête graphique. On établit la connexion de ces deux tables, les champs que l'on souhaite voir apparaître et dans l'onglet Jointure, on définit la jointure qui nous intéresse. On voit tout de fois que dans la partie édition SQL de la requête, la requête est écrite sous la forme d'une clause WHERE et non d'une clause JOIN. Je peux rectifier la chose en remplaçant la syntaxe et on voit que le résultat est toujours le même. Par contre, si avec cette interface pgAdmin, je passe en mode graphique pour revenir en mode éditeur, la requête est à nouveau transformée en une requête WHERE sans garder le mot-clé JOIN.

Notes

Summary

23m 05s

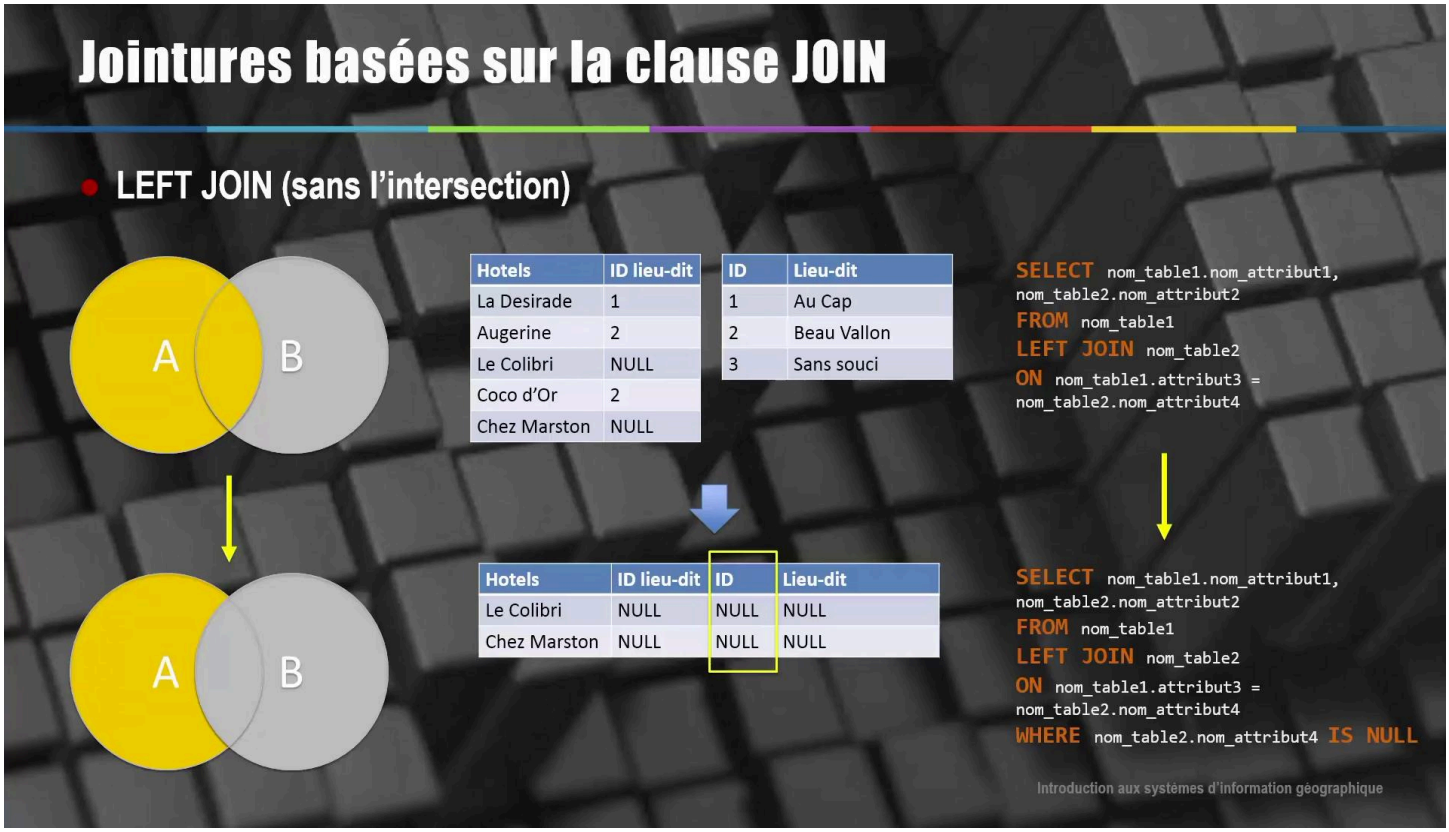


Un autre exemple maintenant avec le logiciel Navicat, qui est un logiciel commercial et qui offre un constructeur graphique de requête SQL intéressant. On ajoute la table des hôtels, la table, cette fois, des lieux-dits. On établit un lien entre le nom de l'hôtel et le nom des lieux-dits. On sélectionne ces deux éléments. On voit que l'on trouve deux hôtels dans le nom est le même que celui des lieux-dits. Dans l'interface graphique, je peux maintenant remplacer la jointure standard par une LEFT JOIN, qui va me donner l'ensemble des 124 hôtels avec en tête de liste les deux pour lequel il existe un lieu-dit dans la jointure. Ensuite, on passe au RIGHT JOIN, qui lui va me donner l'ensemble des lieux-dit, y compris les deux hôtels jointifs. Je crois qu'il y a 346 de ces lieux-dits. Finalement, on peut faire encore un FULL JOIN de ces deux tables pour constater qu'on a dans l'ensemble 466 lignes dans la réponse. Dans ces jointures basées sur la clause JOIN, on peut encore s'intéresser à quelques cas particuliers que l'on peut illustrer, même si l'utilisation de ces diagrammes d'ensemble n'est pas tout à fait correcte, comme on le verra plus tard losqu'on parlera des requêtes de fusion.

Notes

Summary

24m 01s



Ça permet quand même d'illustrer l'esprit du propos. On a la requête LEFT JOIN, dont on aimerait retirer les éléments, qui font le match pour n'avoir que les éléments de la table A qui n'ont pas de correspondance dans la table B. S l'on reprend notre exemple de cinq hôtels et de trois lieux-dits, on se souvient que le LEFT JOIN donnait une table de cinq résultats. Sans l'intersection, cela revient simplement à supprimer les trois cas où la jointure existe, pour ne garder que les deux éléments qui sont non jointifs. Du point de vue de la syntaxe SQL, cela signifie simplement que l'on va ajouter à la requête de jointure, une clause de filtre conditionnel dans laquelle on exprime l'idée que l'attribut de jointure en l'occurrence l'attribut 4 de la table 2, donc le champ ID de la table des lieux-dits, est nul. Même chose pour la jointure droite, avec dans ce cas les quatre champs sélectionnés qui lorsque l'on retire les éléments jointif se réduisent à un seul résultat. À nouveau, en termes de syntaxe SQL, l'adjonction d'une clause conditionnelle basée sur le fait que l'attribut de la jointure dans la première table, est nul. Pour la requête complète, on avait six résultats.

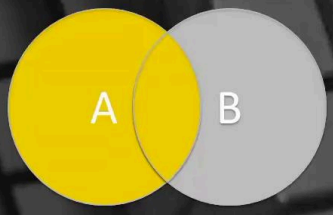
Notes

Summary

25m 29s

Jointures basées sur la clause JOIN

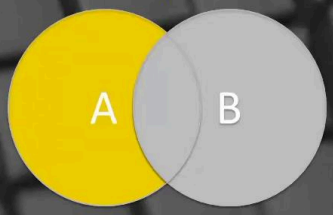
- LEFT JOIN (sans l'intersection)



Hotels	ID lieu-dit
La Desirade	1
Augerine	2
Le Colibri	NULL
Coco d'Or	2
Chez Marston	NULL

ID	Lieu-dit
1	Au Cap
2	Beau Vallon
3	Sans souci

```
SELECT nom_table1.nom_attribut1,
nom_table2.nom_attribut2
FROM nom_table1
LEFT JOIN nom_table2
ON nom_table1.attribut3 =
nom_table2.nom_attribut4
```



Hotels	ID lieu-dit	ID	Lieu-dit
Le Colibri	NULL	NULL	NULL
Chez Marston	NULL	NULL	NULL

```
SELECT nom_table1.nom_attribut1,
nom_table2.nom_attribut2
FROM nom_table1
LEFT JOIN nom_table2
ON nom_table1.attribut3 =
nom_table2.nom_attribut4
WHERE nom_table2.nom_attribut4 IS NULL
```

Introduction aux systèmes d'information géographique

Donc on enlève les trois éléments jointifs pour obtenir un résultat final qui comprend trois objets. Du point de vue SQL, on ajoute cette fois une clause conditionnelle, qui exprime le fait que dans les deux attributs de la jointure, l'un ou l'autre doit être nul.

Notes

Summary

27m 04s