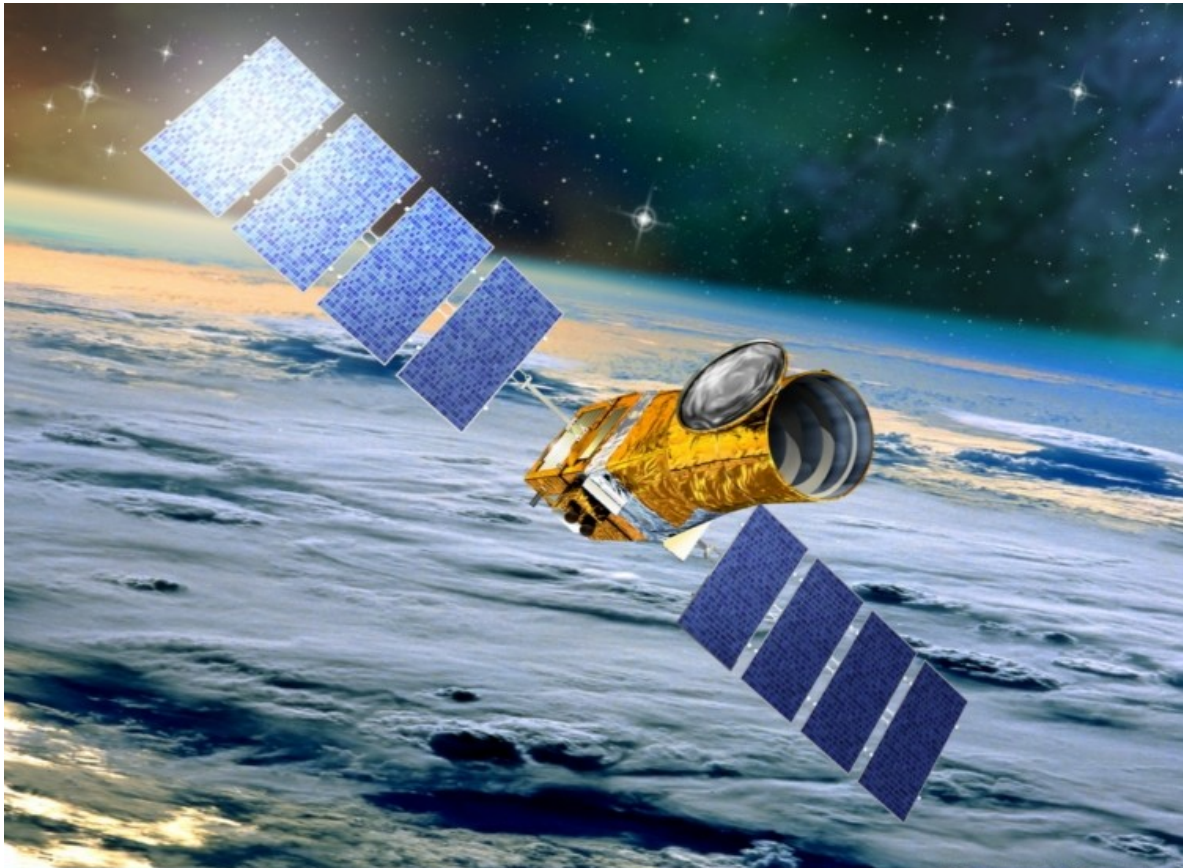


Centre National d'études Spatiales



Etude des bases de données spatialisées

Rapport d'étude

Historique des révisions

VERSION	DATE	OBJET DE LA RÉVISION
V1	21/11/2006	version initiale
V2	07/02/2007	Première relecture: correction de fautes + nouveaux chapitres
V3	20/02/2007	Ajouts de nouveaux chapitres: restrictions, bugs oracle, etc.

	Rédaction	Vérification	Approbation
NOM	H18N181820LNicolas Ribot	Michel-Marie MAUDET	Michel-Marie MAUDET
FONCTION	Expert géomatique	Directeur Général Adjoint	Directeur Général Adjoint
Visa			

Documents de référence

ORDRE	INTITULÉ DU DOCUMENT	AUTEUR
1	Fiche d'évaluation de MySQL	Linagora / Nicolas Ribot
2	Fiche d'évaluation de PostgreSQL / PostGIS	Linagora / Nicolas Ribot
3	Fiche d'évaluation d'Oracle Spatial	Linagora / Nicolas Ribot
4	Fiche d'évaluation d'Oracle Locator	Linagora / Nicolas Ribot
5	Requêtes SQL de test	Nicolas Ribot
6	Présentation de l'étue sous forme de planches	Nicolas Ribot

Table des matières

1.Introduction.....	6.
1.1.Documents de référence.....	6
1.2.Grille de lecture.....	6
2.But de l'étude.....	7
2.1.La problématique du CNES.....	7
2.2.Les éléments de l'étude.....	7
3.Données de test.....	8
3.1.Schéma des bases de données.....	8
3.1.1.Note concernant la table des pays.....	10
3.2.Création et chargement des données spatiales.....	10
3.2.1.Données des emprises d'images.....	10
3.2.2.Données des pays.....	11
3.2.3.Données des masques de nuages.....	11
3.3.Scripts d'installation et scripts SQL.....	12
4.Conditions de l'étude.....	12
4.1.Systèmes de base de données testés.....	12
4.2.Configuration de la machine.....	13
4.3.Accès à la machine.....	13
4.4.Requêtes SQL.....	13
4.5.Mesure du temps d'exécution des requêtes.....	13
4.6.Note sur la fonction de cache dans MySQL.....	15
5.Normes géographiques - Modèle objet.....	15
5.1.OGC.....	15
5.1.1.Le modèle objet.....	15
5.2.SQL MM.....	16
5.3.MySQL.....	16
5.4.PostGIS.....	17
5.5.Oracle.....	17
6.Index spatiaux.....	18
6.1.MySQL.....	19
6.1.1.création/suppression.....	19
6.1.2.Usages.....	20
6.2.PostGIS.....	20
6.2.1.Création/suppression.....	21
6.2.2.Usages.....	22
6.3.Oracle.....	22
6.3.1.Création/suppression.....	23
6.3.2.Usages.....	23
7.Systèmes de Référence spatiale.....	24
7.1.MySQL.....	24
7.2.PostGIS.....	24
7.2.1.Gestion des coordonnées géocentriques.....	25
7.2.2.Gestion des SRS.....	25
7.3.Oracle.....	27
7.3.1.Gestion des coordonnées géocentriques.....	27
Restrictions.....	28
7.3.2.Gestion des SRS.....	28
8.Prédicats spatiaux.....	29

8.1. Notes sur la fonction Relate.....	30
8.2. MySQL.....	31
8.3. PostGIS.....	31
8.4. Oracle.....	31
9. Fonctions spatiales.....	33
9.1. Fonctions d'information.....	33
9.2. Fonctions de création, Opérateurs spatiaux.....	34
9.2.1. MySQL.....	34
9.2.2. PostGIS.....	34
9.2.3. Oracle.....	34
10. Import/Export.....	34
10.1. MySQL.....	35
10.2. PostGIS.....	35
10.3. Oracle.....	36
11. Fonctions diverses.....	37
11.1. PostGIS.....	37
11.2. Oracle Spatial.....	40
12. Requêtes SQL de test.....	40
12.1. Requêtes d'information.....	41
12.2. Prédicats spatiaux sur les bbox.....	41
12.2.1. Problème sur Oracle ?.....	41
12.3. Prédicats spatiaux sur les objets.....	43
12.4. Opérateurs spatiaux.....	43
12.4.1. Résultats graphiques.....	44
13. Paramètres de performance.....	48
13.1. Remplissage des tables.....	48
13.2. MySQL.....	48
13.3. PostGIS.....	48
13.3.1. Données volumineuses.....	49
13.3.2. Réordonner les lignes de la table.....	49
13.4. Oracle.....	50
13.4.1. Package SDO_TUNE.....	50
14. Portage des requêtes spatiales.....	50
14.1. Créer une surcouche applicative.....	51
14.2. Utiliser un outil de correspondance Objet/Relationnel.....	52
15. Restrictions.....	53
15.1. MySQL.....	53
15.2. PostGIS.....	53
15.3. Oracle.....	54
16. Accès externes aux systèmes.....	54
16.1. MySQL.....	54
16.2. PostGIS.....	55
16.3. Oracle.....	55
17. Outils d'administration.....	56
17.1. MySQL.....	56
17.1.1. mysqldump.....	56
17.1.2. mysqlimport.....	57
17.2. PostGIS.....	57
17.2.1. pg_dump, pg_restore.....	57
17.2.2. Slony-I.....	57
17.3. Oracle.....	57
18. Conclusion.....	58

19. Annexe 1: Information sur les systèmes.....	59
19.1. MySQL.....	59
19.2. PostGIS.....	59
19.3. Oracle Spatial.....	59
19.4. Autres ressources.....	59

Index des Illustrations

Illustration 1: Exemple de quad-tree. Les objets spatiaux sont les points noirs cerclés de rouge.....	19
Illustration 2: Exemple de R-Tree.	21
Illustration 3: Exemple de R-Tree.....	23
Illustration 4: Table de métadonnées spatial_ref_sys.....	26
Illustration 5: Polygone de la France en Lambert II Etendu (projection plane).....	27
Illustration 6: Polygone de la France en Longitude Latitude (WGS84).....	27
Illustration 7: Touches() renvoie faux pour ces deux polygones.....	30
Illustration 8: Touches renvoie vrai pour ces polygones.....	30
Illustration 9: bbox du Danemark (en jaune) et de la fenetre de requête (en bleu).....	42

Index des Tables

Table 1: Documents de référence.....	6
Table 2: Matrice des relations spatiales.....	30
Table 3: Correspondance OGC-Oracle des opérateurs.....	32
Table 4: Objets graphiques issus des requêtes spatiales.....	47

1. Introduction

Ce document présente les résultats de « l'étude des bases de données spatialisées » réalisée par LINAGORA et CampToCamp pour le compte du CNES.

Cette étude a été menée sur un serveur proche de ceux utilisés au CNES, avec un jeu de données simulant celui qui résultera de l'exploitation des satellites PLEIADES.

Dans le chapitre « But de l'étude », cette étude sera décrite plus en détail, puis les conditions de l'étude (machines, données) seront décrites dans le chapitre « Conditions de l'étude ».

Le chapitre « Normes Géographiques - Modèle Objet » décrira alors la norme géographique commune aux 3 systèmes testés et le fonctionnement des index spatiaux sera expliqué au chapitre « Index Spatiaux ».

Les résultats de l'étude seront alors présentés dans le chapitre « analyse des résultats ».

1.1. Documents de référence

Ce document se base sur les fiches de description synoptiques des 3 systèmes, sur le document de synthèse des résultats des requêtes SQL de test et sur le document de l'Open Geospatial Consortium définissant une norme de stockage et de manipulation des données géographiques dans une base de données :

Description	Nom du fichier	Référence dans ce document
Fiche synoptique de MySQL	MySQL-5.1beta.ods	FICHE1
Fiche synoptique de PostgreSQL-PostGIS	PostgreSQL-8.1.4_postgis-1.1.6.ods	FICHE2
Fiche synoptique d'Oracle Spatial	Oracle-10gR2_spatial.ods	FICHE3
Fiche synoptique d'Oracle Locator	Oracle-10gR2_locator.ods	FICHE4
Résultats des requêtes SQL	requetes_sql.ods	DOC_SQL
Norme OGC « Simple Feature Specifications for SQL »	opengis_specs_sql_99-049.pdf	NORME_OGC

Table 1: Documents de référence

1.2. Grille de lecture

Ce document détaille les grands chapitres présents dans les fiches d'évaluation des systèmes

(documents FICHE1, FICHE2, FICHE3, FICHE4) et présente également les requêtes SQL de test décrites dans le document DOC_SQL

Pour une bonne compréhension, il est nécessaire de lire également ces fiches, en parallèle de la lecture du présent document.

Le Chapitre 12.4.1 « Résultats Graphiques », notamment, présente de façon graphique les résultats de certaines requêtes SQL détaillées dans le document DOC_SQL.

Les grands groupes de fonctionnalités (projections, index, opérateurs, etc.) sont traités pour chaque système testé (MySQL, PostGIS, Oracle)

2. But de l'étude

L'étude des bases de données spatialisées a pour but de **comparer 3 systèmes de gestion de bases de données** offrant la capacité de stocker et de gérer de l'information géographique: **MySQL, PostgreSQL** et la cartouche PostGIS, **Oracle** et les cartouches spatiales Locator (fournie par défaut avec Oracle) et Spatial sous la forme d'une extension payante.

2.1. La problématique du CNES

Le Centre National d'Etudes Spatiales, le CNES, utilise plusieurs systèmes de gestion de bases de données (SGBD) dans ses différents services et ses différents projets :

- MySQL pour des applications Web par exemple
- Oracle et PostgreSQL pour des applications SIG

Le CNES a également fait le constat que les solutions logicielles Open Source et notamment celles concernant les SGBD ont désormais atteint un niveau de maturité satisfaisant, comme le montre les offres de support et de maintenance disponibles sur le marché.

Le CNES a lancé depuis quelques années le programme PLEIADES visant à remplacer, à terme, les satellites SPOT par une nouvelle génération de satellites haute résolution.

Cette constellation de deux satellites agiles couvrant un champ de 20km, en images optiques couleurs à 70cm de résolution au sol, sera utilisé à la fois pour des applications civiles et militaires, dans les domaines de la défense et sécurité, de la gestion des risques, de la cartographie, de l'agriculture, de la surveillance côtière, de l'étude des forêts, etc...

L'exploitation de ce programme va générer de grandes quantités d'images satellites et mettre à disposition des clients potentiels de ces images un catalogue référençant les images disponibles. Ce catalogue s'appuiera sur une base de données spatiale pour réaliser les requêtes de recherche d'image en fonction de zones géographiques.

La volumétrie des images que devra référencer ce catalogue est d'environ 483 000 images par an et de 5 millions d'albums au total.

L'enjeu pour le CNES est de **rationaliser les moyens d'exploitation**, en particuliers sur les aspects bases de données, en connaissant mieux les SGBD mis en œuvre au sein du Centre et **en analysant les fonctionnalités et la portabilité des requêtes spatiales** entre les systèmes.

2.2. Les éléments de l'étude

En prenant comme cadre de travail la partie Catalogue du projet PLEIADES, c'est à dire en reproduisant le schéma de base de données prévu pour le catalogue et en générant des données

de test, cette étude vise à réaliser les points suivants :

- Montrer les différences entre les cartouches spatiales de MySQL, PostgreSQL et Oracle en terme de fonctionnalité et d'interface,
- Vérifier les fonctionnalités couvertes actuellement par MySQL vis à vis des besoins d'un projet comme PLEIADES HR,
- Analyser la portabilité de requêtes spatialisées PostgreSQL vers MySQL et Oracle vers MySQL.

Les résultats de cette étude seront mis en forme dans différents documents, dont celui-ci :

- Fiches de matrice de fonctionnalités pour chaque système étudié, comprenant une grille d'analyse des SGBD et la couverture fonctionnelle du système ;
- Tableau des requêtes spatiales effectuées sur les systèmes, avec temps d'exécution et remarques ;
- Le présent document de rapport de l'étude ;
- Une présentation de synthèse sous forme de planches.

3. Données de test

Cette étude a été menée sur des données géographiques reproduisant celles qui seront mises en œuvre dans la partie catalogue du projet PLEIADES.

Il s'agit d'un schéma de bases de données, de données géographiques de test, et de scripts SQL permettant d'installer les jeux de données sur chaque système et de tester les différentes fonctionnalités de chaque cartouche spatiale.

3.1. Schéma des bases de données

Le CNES a mis à disposition de l'étude le schéma de base de données pressenti pour la partie catalogue du projet PLEIADES. Ce schéma a été adapté pour chacun des 3 systèmes testés (ajustement des types des colonnes, syntaxes particulières à chaque système, etc.)

La figure suivante montre le schéma SQL mis en place sur les 3 systèmes pour réaliser les tests :

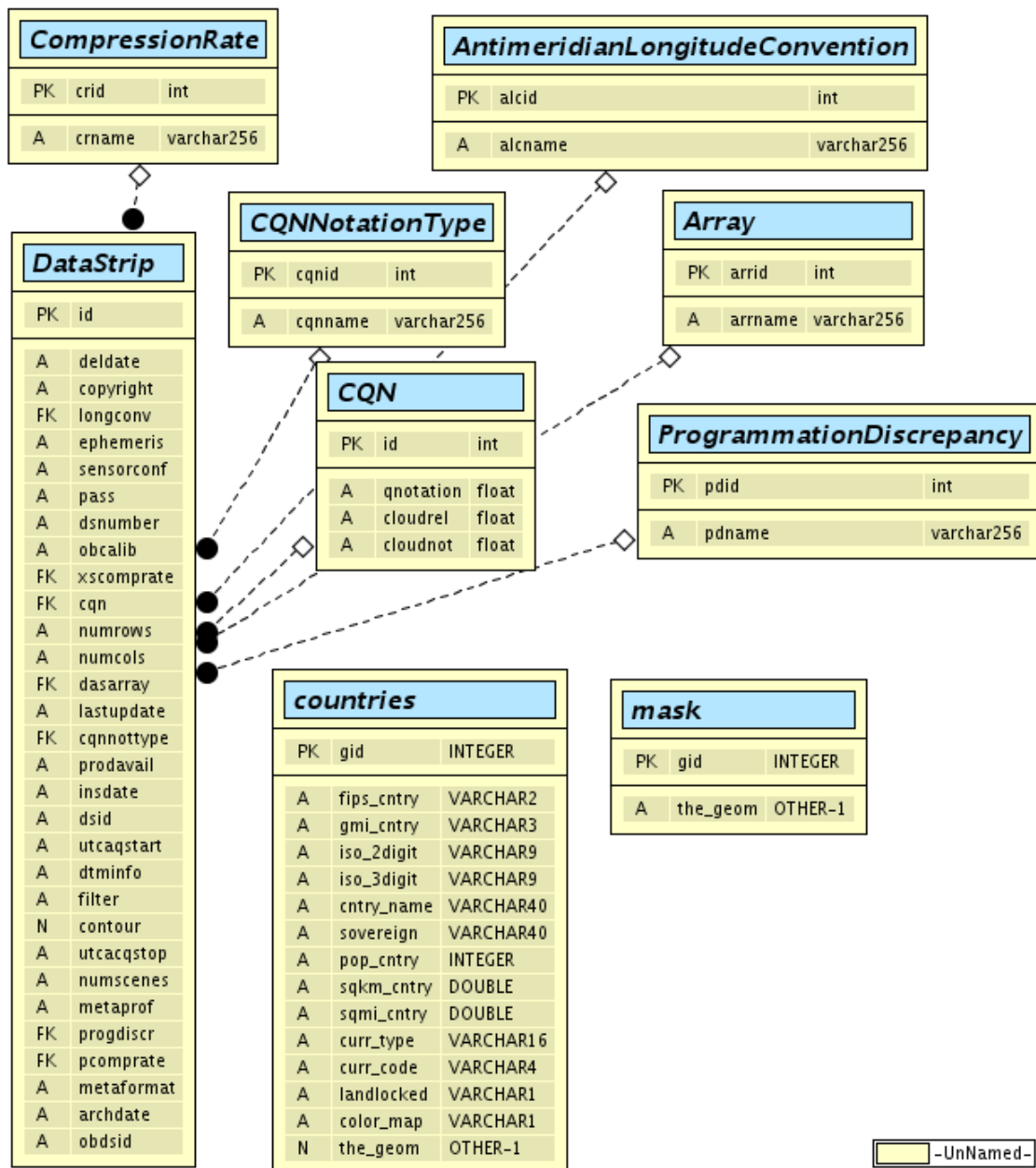


Figure 1: Schéma de la base de données

Ce schéma se compose de tables propres au schéma PLEIADES et de deux tables annexes contenant des données géographiques utilisées pour les requêtes géographiques complexes.

Les tables du schéma PLEIADES sont:

- **Datastrip** : table stockant les emprises des images prises par PLEIADES, aussi appelés **segments**, sous la forme du polygone de l'emprise (champ contour) et les attributs de ces prises de vue.

- *Compressionrate*, *antionlongconv*, *arr*, *cqnotationtype*, *progdiscrepancy* et *cqn* : des tables annexes reliées à la table *datastrip* mais ne contenant pas de données spatiales. Ces tables ne sont pas directement utilisées dans les tests SQL mais sont intégrées au schéma pour reproduire le plus fidèlement le schéma qui sera mis en place pour le catalogue de PLEIADES, notamment les liens entre ces tables et la table centrale *datastrip*.

Les tables annexes sont :

- *countries* : table stockant les multipolygones des pays du monde utilisée pour faire des croisements géographiques avec la table *datastrip*.
- *mask* : table stockant des polygones représentant les masques de nuage, c'est à dire les polygones de la couverture nuageuse de certaines emprises d'images.

3.1.1.Note concernant la table des pays

Devant le temps pris par certaines requêtes spatiales lors du croisement de cette table *countries* avec la table *datastrip*, une deuxième table des pays a été mise en place sur chaque système, ne contenant que certains pays de la table *countries* (les pays les moins étendus).

Cette table s'appelle *cs*, à la même structure que la table *countries*, mais ne contient que 207 multipolygones au lieu des 251 de la table *countries*.

Elle est utilisée dans les requêtes SQL prenant du temps, à la place de la table *countries*.

Le paragraphe suivant décrit plus en détail les données incluses dans ces tables.

3.2.Création et chargement des données spatiales

Les données spatiales de test proviennent de différentes sources. Les paragraphes suivant détaillent la provenance des données.

Pour charger les données dans les différents systèmes, il a été décidé d'utiliser les moyens fournis avec chaque système, dans la mesure du possible.

Oracle et PostGIS fournissent tous deux un utilitaire permettant de charger des données au format Shapefile. MySQL ne fournit aucun programme et recommande de charger les données sous forme de requêtes SQL, avec le format WKT pour les objets géographiques.

3.2.1.Données des emprises d'images

Les emprises des images stockées dans la table *datastrip* ont été générées à partir d'un programme mis à disposition par le CNES, programme générant des polygones au format WKT de façon aléatoire entre des latitudes fixées par paramètre. Le nombre de segments créés est également paramétrable. Ce programme génère un format compatible avec l'import de données dans PostGIS.

Pour chaque système, les données ont été chargées de la manière suivante :

- **PostGIS** : Le fichier généré par le programme a été lancé sur PostgreSQL sur la table *datastrip*, la remplissant des emprises des images.
- **MySQL** : aucun programme livré avec MySQL ne permet de charger des données

spatiales.

Pour pouvoir charger les données des emprises, une procédure stockée écrite en PL/PgSQL a été développée sous PostgreSQL.

Cette fonction permet de générer les requêtes MySQL à partir de la table *datastrip* stockée dans PostGIS.

Un script bash charge alors ce fichier de requêtes dans la base MySQL, après avoir créé la structure du schéma SQL.

- **Oracle** : Les données de la table *datastrip* sont écrites au format Shapefile grâce au programme PostGIS *pgsql2shp*. Ce fichier est ensuite chargé dans Oracle à l'aide d'un programme Java fourni avec l'installation d'Oracle.

Le jeu de données de test comporte environ 1000 polygones (quelques uns ont été écartés du fait de leur coordonnées invalides)

3.2.2. Données des pays

Les polygones des pays sont issus du jeu de données gratuites Digital Chart of the World (<http://www.maproom.psu.edu/dcw/>). Le but de ce jeu de données n'était pas sa précision administrative, mais sa capacité à fournir des polygones complexes de toutes tailles repartis sur l'ensemble du globe.

Le fichier original, au format Shapefile, a servi de base à la table des pays. Suivant les systèmes, les données ont été chargées de la façon suivante:

- **PostGIS** : le fichier Shapefile a été chargé dans PostGIS avec l'utilitaire *shp2pgsql* fourni avec PostGIS. Cet utilitaire crée la table, l'enregistre dans les métadonnées et crée son index. C'est un outil indispensable pour charger des données dans PostGIS.
- **MySQL** : aucun programme livré avec MySQL ne permet de charger des données spatiales. Les exemples donnés dans la documentation insèrent les polygones par leur représentation textuelle.
Pour pouvoir charger les données de pays, une procédure stockée écrite en PL/PgSQL a été développée sous PostgreSQL, pour générer les requêtes MySQL à partir de la table des pays stockée dans PostGIS.
Un script bash charge alors ce fichier de requêtes dans la base MySQL, après avoir créé la structure du schéma SQL.
- **Oracle** : un programme Java est fourni avec Oracle permettant de charger un fichier Shapefile dans le format Oracle 10g. ce programme a été compilé sur la plateforme de test et a été inclus dans un script bash chargeant les données des pays dans Oracle.

La table *cs* utilisée à la place de la table des pays pour les requêtes spatiales les plus longues a été créée par script SQL, en copiant une partie de la table *countries* dans la table *cs*.

3.2.3. Données des masques de nuages

Des masques de nuages ont été dessinés sur quelques segments grâce au logiciel OpenJump. Ces données ont été sauvegardées au format Shapefile, puis insérées dans les 3 systèmes de la façon suivante :

- **PostGIS** : le fichier Shapefile a été chargé dans PostGIS avec l'utilitaire *shp2pgsql* fourni avec PostGIS. Cet utilitaire crée la table, l'enregistre dans les métadonnées et crée son index.
- **MySQL** : aucun programme livré avec MySQL ne permet de charger des données

spatiales. Les exemples donnés dans la documentation insèrent les polygones par leur représentation textuelle.

Pour pouvoir charger les données des masques de nuages, une procédure stockée écrite en PL/PgSQL a été développée sous Postgresql, pour générer les requêtes MySQL à partir de la table des pays stockée dans PostGIS.

Un script bash charge alors ce fichier de requêtes dans la base MySQL, après avoir créé la structure du schéma SQL.

- **Oracle** : un programme Java est fourni par défaut permettant de charger un fichier Shapefile dans Oracle, dans le format Oracle 10g. Les données de la table de masque ont été chargées avec ce programme dans Oracle.

3.3. Scripts d'installation et scripts SQL

L'installation des bases de données dans les 3 systèmes se fait grâce à des scripts Linux chargeant les données grâce à des scripts SQL.

Chaque système met à disposition un programme en ligne de commande permettant de le piloter et d'exécuter des commandes SQL en direct ou stockées dans des fichiers :

- Pour MySQL, il s'agit du programme « mysql » et du script shell `create_db_mysql.sh`
- Pour Oracle, il s'agit du programme « sqlplus » et du script shell `create_db_oracle.sh`
- Pour PostgreSQL, il s'agit du programme « psql » et du script shell `create_db_pg.sh`

Chaque système possède ainsi son script de contrôle et ses scripts SQL et de données permettant de créer l'environnement de test. La structure du script de contrôle est la même pour les 3 systèmes, avec comme étapes :

1. création du schéma de base de données à partir d'un script SQL ;
2. Insertion des données de la table des emprises des images (*datastrip*) (shapefile ou script SQL) ;
3. Suppression des données d'emprise invalide, par script SQL ;
4. Création et population de la table des pays (*countries*) (shapefile ou script SQL) ;
5. Création et population de la table des masques de nuages (*mask*) (shapefile ou script SQL) ;
6. Création et population de la table *cs*, extrait de la table des pays ;
7. Pour PostgreSQL uniquement: création des procédures stockées permettant de générer les fichiers de données au format SQL (cf. paragraphes précédents) ;

Ces scripts sont livrés dans une archive en même temps que les documents de l'étude.

4. Conditions de l'étude

4.1. Systèmes de base de données testés

L'étude des bases de données spatialisées a été conduite sur les SGBD suivants :

- **MySQL 5.1.12 beta** ;
- **PostgreSQL 8.1.4** avec cartouche spatiale PostGIS 1.1.6, support GEOS 2.2.1, PROJ4

4.4.9 ;

- Oracle 10g Release 2 (10.2) avec cartouches spatiales Locator et Spatial ;

Ces 3 systèmes ont été installés sur la machine de test, dans la configuration normale d'installation (aucune personnalisation n'a été faite sur les systèmes après installation).

L'utilisateur de test a été ajouté comme administrateur de chacun des systèmes, pour permettre de se connecter et de réaliser les opérations de création du schéma.

Lors des tests, les 3 systèmes de gestion de bases de données étaient démarrés (paramétrés pour un démarrage automatique avec le serveur).

4.2. Configuration de la machine

La machine mise en place pour les tests entre les 3 systèmes est la suivante:

- OS : Red Hat Linux Enterprise ES 4
- Processeur : XEON 5050 3.0GHZ/2X2MB 667FSB
- Mémoire : 2GB FB 667MHZ
- Disques : 2 x 146GB SAS (10, 000RPM)
- Contrôleur raid : PERC 5/I intégré

Cette configuration est proche de celle des serveurs utilisés par le CNES, comme indiqué lors des points techniques.

4.3. Accès à la machine

L'accès à la machine de test s'est fait par SSH, à distance. Les scripts de création du schéma, de chargement des données, des requêtes SQL de test ont été copiés sur le serveur par SCP, puis lancés en local grâce à l'accès SSH.

4.4. Requêtes SQL

Les requêtes SQL de test ont été exécutées sur les 3 systèmes à partir d'une console SSH connectée à la machine de test.

Chaque système propose un programme en ligne de commande permettant d'exécuter tout type de requêtes sur le systèmes, de sauver le résultats des requêtes dans des fichiers, ou de d'exécuter des requêtes depuis des fichiers :

- Pour MySQL, il s'agit du programme mysql ;
- Pour PostGIS, il s'agit du programme psql ;
- Pour Oracle, il s'agit du programme sqlplus ;

4.5. Mesure du temps d'exécution des requêtes

Dans la fiche de synthèse des requêtes SQL de test, des durées d'exécution des requêtes sont indiqués pour les 3 systèmes. Ces durées sont données a titre indicatif, pour faire une première comparaison entre les systèmes.

Il ne s'agit pas de tests de performance, ou chaque requête aurait été lancée plusieurs fois pour en analyser son temps moyen d'exécution.

Ainsi, ces temps peuvent varier un peu lorsque la même requête est lancée plusieurs fois.

Pour chaque système, l'activation de la mesure du temps de requêtes nécessite une commande particulière indiquée ci-après:

- MySQL : les temps d'exécution sont automatiquement indiqués en secondes dans la console mysql. Par exemple :

```
mysql> select count(*) from countries;
+-----+
| count(*) |
+-----+
|      251 |
+-----+
1 row in set (0.01 sec)
```

- Oracle (Spatial et Locator) : les temps d'exécution sont indiqués en heure:minutes:secondes après avoir lancé la commande suivante dans la console sqlplus:

```
set timing on;
```

Par exemple :

```
SQL> select distinct c.the_geom.st_isvalid() from cs c;

C.THE_GEOM.ST_ISVALID()
-----
                        1
                        0

Elapsed: 00:00:02.56
```

- PostgreSQL : les temps d'exécution sont indiqués en millisecondes après avoir lancé la commande suivante dans la console psql:

```
\timing
```

Par exemple :

```
test_spatial=# select count(*) from countries;
count
-----
      251
(1 row)

Time: 1.960 ms
```

4.6.Note sur la fonction de cache dans MySQL

MySQL, lors d'une installation par défaut sur la distribution Linux installée sur la machine de test, offre un mécanisme de cache des requêtes SQL : Tout résultat d'une requêtes SQL d'interrogation est cachée par le système et renvoyé ensuite quand cette même requête est ré-exécutée. Les temps indiqués pour l'exécution d'une requête sont alors faux si cette requête a déjà été exécutée.

Cette fonction de cache a été désactivée dans la console mysql avant de lancer les requêtes de test, en exécutant la commande MySQL suivante :

```
SET SESSION query_cache_type = OFF;
```

Les autres systèmes ne proposent pas cette fonction de cache par défaut.

5.Normes géographiques - Modèle objet

5.1.OGC

Une première analyse des spécifications des 3 cartouches spatiales fait apparaître que les 3 systèmes se basent sur la norme « OGC OpenGIS® Simple Features Specification For SQL Revision 1.1 ». document « OpenGIS Project Document 99-049 » pour les types géographiques et les fonctions agissant sur ces types.

Dans le reste de ce document, le terme *norme OGC* désigne ce document 99-049.

Cette norme définit une structure pour les types géométriques et décrit les fonctions agissant sur ces types.

Il a été décidé de comparer ces 3 systèmes face à cette cette norme, en incluant dans la comparaison les fonctions fournies par les systèmes qui ne sont pas décrites dans la norme.

Le but de ce chapitre n'est pas de décrire en détail cette norme. Pour une description complète de celle-ci, se référer au document de spécifications cité ci-dessus, disponible sur le site: <http://www.opengeospatial.org/>.

5.1.1.Le modèle objet

La norme OGC définit des types géométriques pour représenter les objets : points, lignes, polygones, multi-objets. Les types disponibles sont :

- GEOMETRY
- POINT
- LINestring
- POLYGON
- GEOMCOLLECTION
- MULTIPOINT
- MULTILINestring
- MULTIPOLYGON

Par défaut, l'interpolation entre les points d'une géométrie est linéaire. Oracle permet le stockage d'arc de cercles comme parties d'une géométrie, PostGIS et MySQL ne le permettent pas.

Les fonctionnalités agissant sur les arc de cercles sont décrites mais non testées dans cette étude.

Pour chacun des types géométriques, la norme OGC définit la notion d'intérieur, d'extérieur et de frontière de l'objet.

Ces notions importantes sont alors utilisées pour qualifier le résultats de prédicats spatiaux (cf. § Prédicats Spatiaux). Il est important de bien comprendre ces notions pour appréhender correctement les prédicats spatiaux et éviter des erreurs d'interprétation des résultats.

5.2. SQL MM

La norme *ISO/IEC 13249 SQL/MM Part 3* est un standard international définissant le stockage et la manipulation de données spatiales en SQL. Il peut se comparer à la norme OGC dans son esprit et est une sous partie d'un ensemble plus vaste décrivant la gestion des données multimédia en SQL.

Oracle Spatiale implémente cette norme en plus de la norme OGC

Pendant le déroulement de cette étude, une nouvelle version de PostGIS, la version 1.2, est sortie. Cette nouvelle version implémente en partie cette norme.

Cette version de PostGIS n'a pas été testée au cours de cette étude.

Cette norme n'est pas prise en compte dans cette étude.

5.3. MySQL

MySQL implémente en partie la norme OGC. Les principales fonctions définies par la norme sont absentes de MySQL.

Le document FICHE1 liste les fonctionnalités définies par la norme OGC et indique pour chacune le niveau de support MySQL.

Dans la version testée, il est possible :

- de stocker des objets géographiques,
- d'indexer ces objets,
- d'utiliser des opérateurs spatiaux sur les rectangles englobant des objets (bbox),
- d'avoir des informations sur les objets spatiaux: type, nombre de points, nombre d'objets dans le cas des collections, etc...,
- de connaître la longueur ou la surface des objets (suivant le type d'objets),
- de stocker un identifiant d'un système de référence spatiale (SRS) dans l'objet.

Il n'est pas possible :

- d'utiliser des fonctions de création d'objets: *buffer*, *convexHull*, *union*, *différence*, etc...,

- d'utiliser des opérateurs spatiaux (within, intersects, disjoint, etc) sur les objets eux-mêmes,
- de reprojeter les données ou gérer les SRS,
- d'administrer des métadonnées associées aux données spatiales,
- d'importer ou d'exporter des données spatiales depuis/vers d'autres formats que le WKT ou WKB.

Les prochaines versions de MySQL devraient implémenter un plus grand nombre de fonctions définies par la norme OGC.

5.4. PostGIS

Depuis la version 1.6, PostGIS est conforme à la norme OGC.

Le modèle objet est implémenté entièrement et toutes les fonctions et opérateurs décrit par la norme sont disponibles, notamment :

- Le respect du nommage des types et fonctions, sauf dans le cas de *union*, mot réservé par SQL et défini par la norme comme la fonction d'union de deux objets spatiaux. PostGIS appelle cette fonction : *geomunion* ;
- Le support pour les systèmes de référence spatiale (SRS), sur la base d'EPSG (cf. § sur les SRS) ;
- Le support de métadonnées associées aux données spatiales stockées en base (nom de la table, nom de la colonne spatiale, type d'objets stockés, SRS) ;
- Le support de fonction complexes manipulant les données spatiales, comme *buffer()*, *convexHull()*, *union()*, *intersection()*, etc...

PostGIS offre également des fonctions non décrites par la norme, comme la gestion des points ou lignes mesurées (portant une information sur chaque segment), des fonctions pour transformer des objets spatiaux (création de lignes à partir de points, de polygones à partir de lignes fermées, etc.).

PostGIS dispose également d'outils d'import/export de format shapefile, très pratique pour charger des données spatiales dans PostGIS ou pour les exporter vers un SIG ne sachant pas gérer PostGIS. Il existe également des fonctions pour exporter les données en GML et KML dans la dernière version (1.2.1).

Dans sa version 1.2, non testée dans cette étude, PostGIS supporte en partie la norme SQL-MM, qui spécifie l'usage des données spatiales dans une base de données relationnelle.

5.5. Oracle

Oracle est conforme à la norme OGC depuis la version 10g. Les versions précédentes étaient conformes à la norme « Simple Features - SQL - Normalized Geometry 1.1 »

Oracle ne respecte pas les règles de nommage concernant les fonctions, en qualifiant le nom de chaque fonction par le préfixe « SDO_ » et par un schéma particulier, variant suivant le type des fonctions : *SDO_UTIL*, *SDO_GEOM*, *SDO_LRS*, etc.

La fonction *convexHull* est par exemple appelée *SDO_GEOM.SDO_CONVEXHULL()*, la fonction *Intersection()* devient *SDO_GEOM.SDO_INTERSECTION()*.

Ces contraintes apportées par le système pénalisent fortement la portabilité des requêtes, car une requête définie en respectant la norme OGC provoque une erreur dans Oracle. Pour assurer une portabilité, il faut donc passer par un système intermédiaire réécrivant les requêtes pour chaque système, en établissant une correspondance entre les fonctions définies dans la norme et leur implémentation dans les systèmes.

Oracle supporte donc :

- La gestion des SRS (basés sur EPSG et sur le système Oracle), le changement de SRS pour les objets spatiaux ;
- Les Opérateurs et fonctions géographiques agissant sur les objets ;
- La gestion des métadonnées des tables spatiales ;
- l'indexation efficace des données spatiales ;

Il existe également un programme écrit en Java, livré avec Oracle, permettant de charger des données au format ESRI Shapefile dans Oracle, avec enregistrement des métadonnées et indexation spatiales.

Oracle va bien au delà de la norme OGC en offrant toute une série de fonction spatiales permettant de gérer tous les aspects des systèmes d'information géographiques, notamment:

- gestion des coordonnées géocentriques, avec calcul dans un système sphérique ;
- Fonction de simplification et de validation des objets ;
- Gestion des cercles et arcs de cercle comme objets géographiques. Un polygone pouvant être constitué de parties rectilignes et d'arcs de cercle. (sauf pour les données en coordonnées géocentriques) ;
- Gestion des réseaux, des graphes ;
- Calcul d'itinéraires ;
- Gestion des rasters (stockage, transformation de formats, opérations, etc.) ;
- Géocodage ;
- Accès par services web ;
- Data mining (recherche d'information agrégée de façon automatique, dans le domaine spatial) ;

6.Index spatiaux

Les index sont au coeur des systèmes de gestion de bases de données. Ils permettent d'augmenter la performance lors des requêtes en définissant des classements sur certains champs des tables.

Généralement, les données spatiales sont volumineuses et demandent un temps de traitement important. Ces données étant distribuées dans l'espace, il peut être intéressant de les classer dans cet espace et d'utiliser ce classement dans un index.

Les 3 systèmes disposent d'un système d'indexation spatiale et recommandent fortement, ou obligent, d'utiliser l'indexation spatiale pour des colonnes stockant des objets géographiques.

Les paragraphes suivants décrivent les index spatiaux de chacun des systèmes et montrent leur

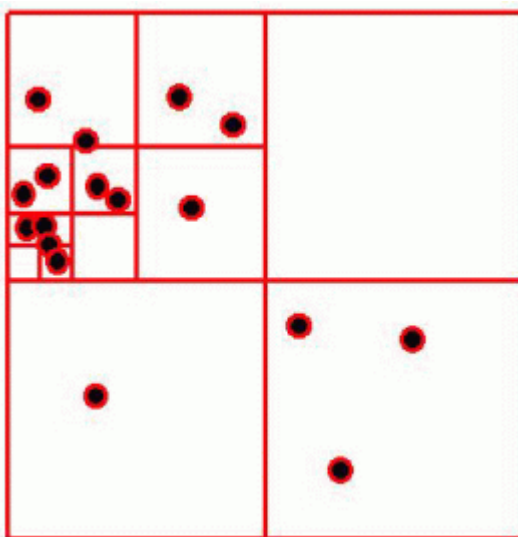
utilisation.

6.1.MySQL

MySQL utilise pour l'indexation des données spatiales un index de type R-Tree (Regional Tree) avec séparation quadratique de l'espace. (également appelé quad-tree)

L'index de type R-Tree divise l'espace en secteurs pour associer un certain nombre d'objets à chaque secteur. Lorsque le nombre d'objets maximum défini pour un secteur est atteint, ce secteur est subdivisé en 4 parties égales à son tour (séparation quadratique).

Le schéma suivant montre un index de type Quad-Tree :



*Illustration 1: Exemple de quad-tree.
Les objets spatiaux sont les points noirs cerclés de rouge*

MySQL index les données spatiales en prenant en compte leur rectangle englobant (bounding box en anglais, ou bbox).

L'indexation des colonnes est facultatif avec MySQL, mais très fortement recommandée tant le gain sur les requêtes spatiales est important, quand le nombre d'objets devient significatif.

6.1.1.création/suppression

La création d'un index spatial dans MySQL peut se faire a trois niveaux:

- avec la commande create table, ex

```
CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g));
```

- Avec la commande alter table:

```
ALTER TABLE geom ADD SPATIAL INDEX (g);
```

- Avec la commande create index, ex:

```
CREATE SPATIAL INDEX sp_index ON geom (g);
```

Il est possible de supprimer les index avec les commandes:

```
ALTER TABLE geom DROP INDEX g;  
ou  
DROP INDEX sp_index ON geom;
```

6.1.2. Usages

L'optimisateur de requêtes ne cherche s'il faut utiliser les index spatiaux que si la clause WHERE de la requête contient un des opérateurs travaillant sur les bounding box, comme MBRContains, MBRWithin, MBRIntersects, etc.

Exemple de requêtes utilisant l'index spatial, si celui-ci a été créé :

```
SELECT fid, AsText (g)  
FROM geom  
WHERE MBRContains (GeomFromText ('Polygon ((30000 15000, 31000  
15000, 31000 16000, 30000 16000, 30000 15000)) '), g)
```

6.2. PostGIS

PostGIS permet l'indexation des colonnes contenant des données spatiales grâce aux index de type GiST (Generalized Search Tree). Ce type d'index permet d'indexer presque tous les types de structures : tableaux d'entiers, données spectrales, etc)

PostgreSQL supporte les index de type R-Tree, mais leur implémentation est moins puissante que celle de type GiST et ne doit donc pas être utilisé pour l'indexation des données spatiales.

Ce type d'index GiST, très performant est générique. Il faut lui fournir un type de recherche. C'est le type R-Tree, bâti au dessus de GiST, qui est utilisé pour indexer les données spatiales dans PostGIS.

L'index GiST offre deux avantages par rapport aux index de type R-Tree présents dans PostGIS:

- Il permet l'indexation de colonnes contenant des valeurs nulles ;
- Il permet d'indexer de gros objets en ne prenant en compte que leur bounding box, ce que ne ferait pas un index R-Tree, limitant l'indexation des objets de moins de 8 Ko.

Le schéma suivant montre une indexation de l'espace de type R-Tree :

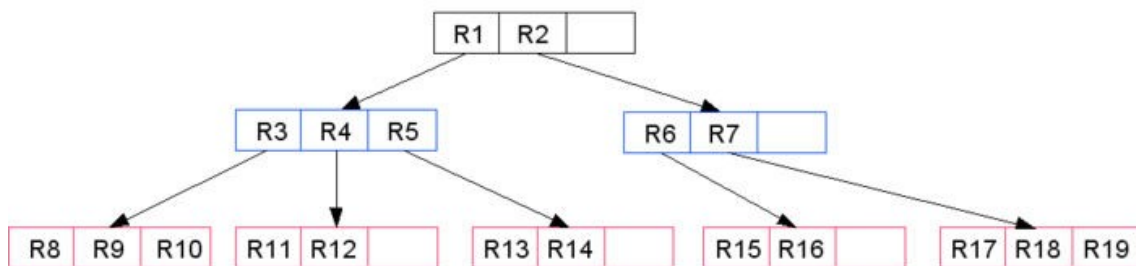
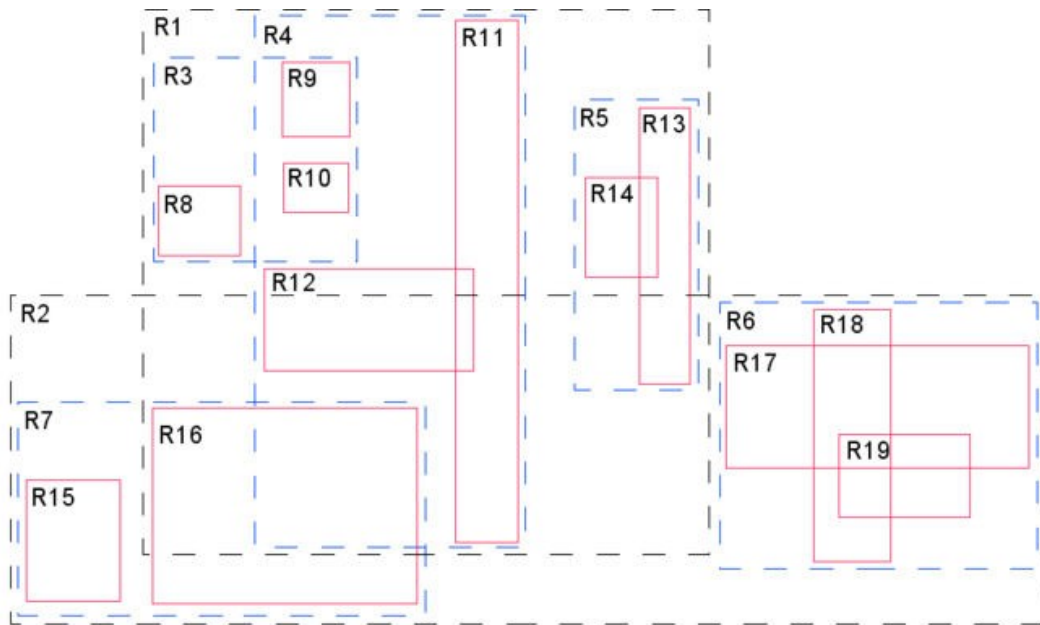


Illustration 2: Exemple de R-Tree.

6.2.1. Création/suppression

La syntaxe pour créer un index sur une colonne géométrique est :

```
CREATE INDEX [indexname] ON [tablename] USING
GIST ( [geometryfield] GIST_GEOMETRY_OPS );
```

La création des index peut prendre du temps. Il est recommandé de créer les index après le chargement des données, si la table doit contenir un grand nombre de données.

De même, pour une utilisation optimale des index, il est recommandé de collecter régulièrement les statistiques sur la table ou la colonne, grâce à la commande suivante:

```
VACUUM ANALYZE [table_name] [column_name];
```

Pour supprimer un index, utiliser la commande:

```
DROP INDEX index_name;
```

6.2.2. Usages

L'optimisateur de requête de PostgreSQL prend en compte l'index spatial que si des opérateurs agissant sur les rectangles englobant sont utilisés, comme les opérateurs :

&& (intersects), @ (contains), ~ (within), etc.

Il est donc important d'inclure de tels opérateurs dans les requêtes SQL spatiales pour être sûr de la prise en compte de l'index dans la requête.

Par exemple, la requête suivante ne va pas utiliser d'index spatial :

```
SELECT the_geom FROM geom_table WHERE distance(
  the_geom, GeomFromText( 'POINT(100000 200000)', -1 ) ) <
  100
```

Alors que celle-ci le fait:

```
SELECT the_geom FROM geom_table WHERE the_geom
  && 'BOX3D(90900 190900, 100100 200100)::box3d AND
  distance( the_geom, GeomFromText( 'POINT(100000 200000)', -1 )
  ) < 100
```

(L'opérateur BOX3D crée un rectangle de 200 unités autour du point cherché.)

Les index de type GiST permettent également d'utiliser le résultat d'une fonction comme index, ce qui peut être très puissant, exemple :

```
CREATE INDEX foo_transformed_idx ON foo USING GIST
(transform(geom, 4326));
```

Dans cet exemple, l'index créé sur la colonne geom de la table foo utilise la fonction transform() pour indexer le résultat du changement de système de coordonnées des objets de la table foo vers le système WGS84 (srid=4326 dans postgis)

6.3. Oracle

Oracle Spatial et Oracle Locator utilisent le même système d'index spatiaux. La description faite dans ce chapitre s'applique donc aux deux systèmes.

Oracle propose deux types d'index pour indexer les données spatiales :

- Un index de type R-Tree (voir le chapitre des index de PostgreSQL pour une description des R-Tree) ;
- Un index de type Quad-Tree (voir le chapitre des index de MySQL pour une description des Quad-Tree) ;

Mais la documentation recommande fortement d'utiliser le type R-Tree, pour des questions de performance, les index Quad-Tree devant disparaître dans les prochaines versions d'Oracle Spatial ou Locator.

Il est obligatoire d'indexer les colonnes contenant des objets spatiaux avant de pouvoir utiliser la plupart des opérateurs spatiaux et certaines fonctions.

Un index de type R-Tree fonctionne en partitionnant l'espace en sous-régions, chacune contenant un certain nombre d'objets au maximum. Quand cette limite est atteinte, l'espace en question est divisé à son tour en sous-partie, et les objets géographiques attachés à ces parties.

La figure suivante, extraite de la documentation Oracle, montre un tel découpage de l'espace et l'organisation de l'index qui en découle :

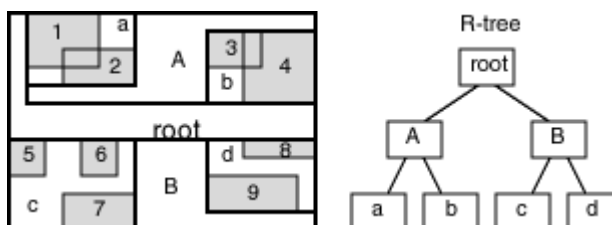


Illustration 3: Exemple de R-Tree

6.3.1. Création/suppression

La syntaxe pour créer un index spatial sur une colonne géométrique est :

```
CREATE INDEX NAME_idx ON my_table (geom_column)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

L'index spatial peut être créé sur 2 (défaut), 3 ou 4 dimensions. Si l'index spatial a plus de deux dimensions, seul l'opérateur spatial *SDO_FILTER* peut être utilisé dans la requête.

Il est possible d'estimer la taille que va prendre un index spatial grâce à la fonction :

```
SDO_TUNE.ESTIMATE_RTREE_INDEX_SIZE
```

Pour supprimer un index, utiliser la commande :

```
DROP INDEX idx_name;
```

6.3.2. Usages

Oracle utilise la notion de filtre primaire et secondaire pour l'indexation des données spatiales. L'usage des deux filtres à la suite permet d'obtenir un résultat unique pour la recherche.

Le filtre primaire agit sur les rectangles englobant des objets et est donc très rapide.

Il s'utilise grâce à la fonction :

```
SDO_FILTER(geom1, geom2, param)
```

Le filtre secondaire agit sur les objets eux-mêmes. Il est mis en œuvre par exemple avec la

fonction qui utilise également un filtre primaire pour éliminer certains candidats :

```
SDO_RELATE(geometry1 SDO_GEOMETRY,  
            geometry2 SDO_GEOMETRY,  
            param      VARCHAR2)
```

Exemple de requête utilisant les index spatiaux, si ceux-ci ont été définis sur les colonnes :

```
select count(d.id)  
from datastrip d, cs c  
where sdo_anyinteract(d.contour, c.the_geom) = 'TRUE';
```

7. Systèmes de Référence spatiale

Les données géographiques représentent des objets à la surface de la terre, qui est à peu près sphérique.

Ces objets sont souvent représentés sur des surfaces planes (cartes ou écrans). Il faut donc les projeter grâce à une formule mathématique, la projection. Les données ainsi transformées sont dans un nouveau système de référence spatiale : SRS.

Il est essentiel de pouvoir changer le systèmes de coordonnées d'un jeu de données lorsqu'on manipule des données spatiales : données provenant de plusieurs sources, donc potentiellement dans plusieurs SRS, édition de cartes dans différentes projections, etc.

Dans ce domaines, les différents systèmes utilisés offrent des fonctionnalités très différentes.

Les paragraphes suivants détaille la gestion des SRS dans les différents systèmes.

Les chapitres sur l'analyse des résultats évoquent également le sujet.

7.1. MySQL

MySQL ne permet pas le changement de SRS. Il ne connaît pas la notion de système de projection. Il permet cependant de stocker un identifiant¹ d'un système de coordonnées (comme un code EPSG²) avec chaque objet, identifiant qui peut être récupérer avec la fonction SRID(geom).

Les calculs sont donc faits dans un espace euclidien (plan).

7.2. PostGIS

PostGIS permet le changement de SRS grâce à la fonction tranform(geom, srid).

PostGIS utilise la bibliothèque C/C++ PROJ4 pour effectuer le changement de SRS.

Les données relatives aux SRS sont stockées dans une table de métadonnées définie par la

-
- 1 Cet identifiant est requis par la norme OGC, c'est le premier pas pour pouvoir supporter le changement de SRS:
 - 2 L'EPSG (European Petroleum Survey Group) maintien une base de données des systèmes de projections utilisés dans le monde. Elle est largement utilisée par la communauté Open Source.

norme OGC (cf. § sur les normes): `spatial_ref_sys`.

Elle contient la définition de plus de 2670 systèmes de projection couvrant le monde entier.

PostGIS ne permet pas de travailler avec des objets ayant différents identifiants de projection. Il faut, au préalable, convertir un jeu de données depuis son SRS vers celui de l'autre jeu.

7.2.1. Gestion des coordonnées géocentriques

Dans PostGIS, les calculs se font en deux dimensions, dans un espace euclidien, quelque soit le SRS. Les calculs de distance et de surface pour les systèmes géocentriques ne sont alors pas pertinents car exprimé en degrés, variant avec la latitude.

Il existe cependant des fonctions travaillant avec des coordonnées géocentriques :

`distance_sphere()`, `distance_spheroid()`, `length2d_spheroid()`, `length3d_spheroid()`, qui permettent de calculer la distance de deux points sur une sphère ou un ellipsoïde (sphéroïde).

Le repère de coordonnées est centré sur la longitude 0 et la latitude 0. Les coordonnées des objets doivent être comprise entre -180 et +180 pour les longitudes, et -90 et +90 pour les latitudes.

Les longitudes exprimées entre 0 et 360 pour ne sont pas correctement interprétées (c'est à dire correspondant a un système sphérique).

Le problème pour gérer les coordonnées géocentriques vient du fait que PostGIS ne connecte pas les points de longitude -180, +180, qui représentent le même point sur la terre.

Par exemple, le polygone suivant n'est pas centré sur l'origine, mais décalé sur la droite :

```
POLYGON ((350 -10, 10 -10, 10 10, 350 10, 350 -10))
```

Il existe, sur le forum PostGIS, des exemples de fonctions/méthodes permettant de convertir les coordonnées d'objets, ou de découper des objets en deux polygones:

<http://postgis.refractions.net/pipermail/postgis-users/2006-June/012483.html>

7.2.2. Gestion des SRS

PostGIS se base sur les systèmes de projection définis par l'EPSG.

L'European Petroleum Survey Group maintient une base de données des systèmes de projections utilisés dans le monde. Elle est largement utilisée par la communauté Open Source.

Il est possible d'ajouter de nouvelles définitions de SRS en insérant les paramètres de la projection, ainsi qu'un code unique, dans la table `spatial_ref_sys`.

Les objets spatiaux stockés dans PostGIS, comme le précise la norme OGC, possèdent un attribut appelé SRID (Spatial Reference Identifier) qui indique le code du SRS associé à cet objet. Ce code³ est une des valeurs contenues dans la table `spatial_ref_sys`.

La valeur par défaut du SRID d'un objet est -1 (pas de SRS). Toute autre valeur doit avoir une entrée correspondante dans la table `spatial_ref_sys`.

Le schéma suivant montre la structure de cette table :

³ Il faut noter que ce code est propre au SGBD. Ce n'est pas une norme. Chaque système est libre de gérer les SRID a sa guise. PostGIS utilise, comme SRID, les code EPSG.

<i>spatial_ref_sys</i>		
PK	srid	INTEGER
A	auth_name	varchar256
A	auth_srid	INTEGER
A	srttext	VARCHAR2048
A	proj4text	VARCHAR2048

Illustration 4: Table de métadonnées spatial_ref_sys

Le tableau suivant indique la signification de chaque champ:

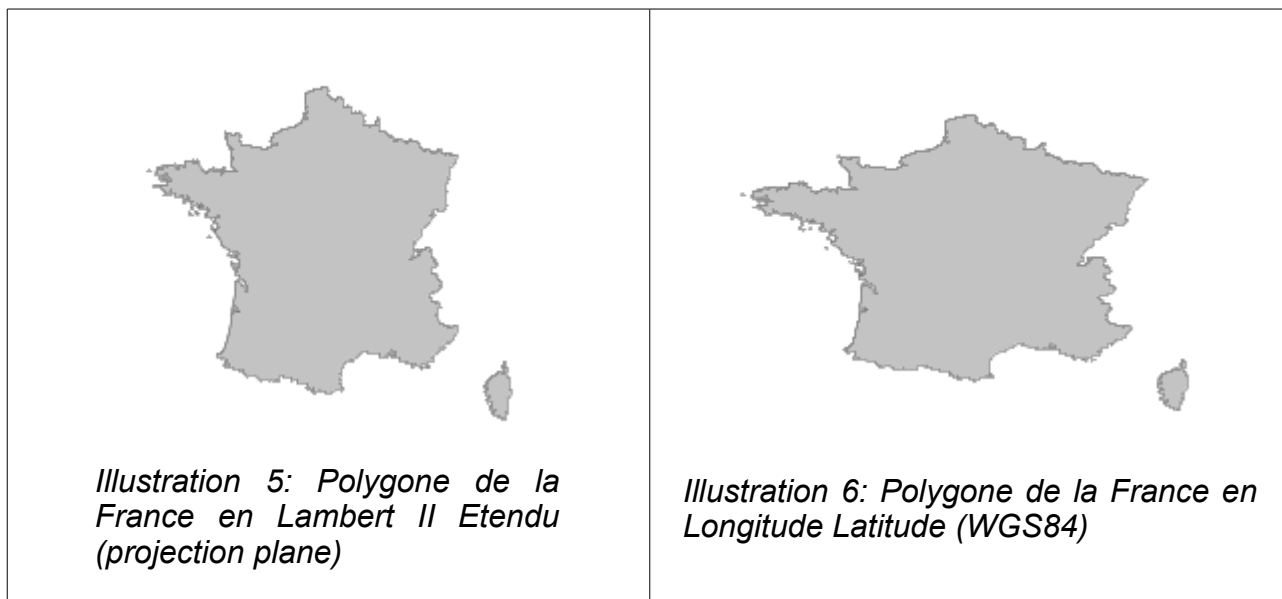
Champs	valeur
srid	L'identifiant unique du SRS. (= code EPSG)
auth_name	Le nom de la norme ou de l'organisme de normalisation cité pour ce SRS
auth_srid	L'identifiant du SRS tel que définit par la norme ou l'organisme
srttext	La représentation WKT du SRS, comme indiqué dans la norme OGC. Par exemple : <pre>PROJCS["NAD83 / UTM Zone 10N", GEOGCS["NAD83", DATUM["North_American_Datum_1983", SPHEROID["GRS 1980",6378137,298.257222101]], PRIMEM["Greenwich",0], UNIT["degree",0.0174532925199433]], PROJECTION["Transverse_Mercator"], PARAMETER["latitude_of_origin",0], PARAMETER["central_meridian",-123], PARAMETER["scale_factor",0.9996], PARAMETER["false_easting",500000], PARAMETER["false_northing",0], UNIT["metre",1]]</pre>
proj4text	La définition de la projection dans le format PROJ4 (PostGIS utilisant PROJ4 pour changer de SRS). Par exemple : <pre>+proj=utm +zone=10 +ellps=clrk66 +datum=NAD27 +units=m</pre>

Grâce à des fonctions de PostGIS, il est possible de changer le SIRD d'un objet, soit directement en lui en affectant un nouveau (pas de changement de l'objet lui-même) (utilisé par exemple quand on veut renseigner un SRS pour un objet dont on ne le connaissait pas avant), soit en changeant son SRS en appelant la méthode :

```
transform(geom, srid);
```

L'objet est alors modifié.

La figure suivante montre le polygone de la France dans deux SRS différents.



7.3.Oracle

Oracle offre un ensemble d'outils et de tables pour gérer les systèmes de Référence spatiale (SRS) à travers le package `SDO_CS` (CS= Coordinate System)

Oracle permet le changement de SRS et gère les systèmes de coordonnées géocentriques. Il permet également de convertir les unités de calculs (m, km, mi, ft, etc.).

Les opérateurs spatiaux définis dans Oracle (`SDO_ANYINTERACT`, `SDO_RELATE`, `SDO_FILTER`, etc) gèrent les objets ayant des SRID différents en projetant la bbox du deuxième objets dans le SRS du premier objet.

Les fonctions spatiales du package `SDO_GEOM` (`SDO_DISTANCE`, `SDO_UNION`, `SDO_INTERSECTION`, `SDO_XOR`, etc), par contre ne peuvent travailler que sur des objets ayant le même SRID.

Etant donnée la richesse des fonctionnalités d'Oracle dans le domaine de la gestion des SRS, ce chapitre ne va pas présenter en détail le système. Pour cela, se reporter au guide d'utilisateur d'Oracle Spatial.

Ce chapitre insiste plutôt sur les grandes fonctionnalités offertes par le système.

7.3.1.Gestion des coordonnées géocentriques

Oracle gère les coordonnées géocentriques des objets spatiaux. Il réalise les calculs dans un système sphérique en tenant compte de la forme de la terre.

Cette fonctionnalité, absente des autres systèmes, est très utile des lors que les objets stockés sont dans un système géocentrique.

Les données géocentriques doivent être indexées avec un index spatial de type géocentrique, ce qui est automatiquement fait lors de l'indexation de données géocentriques. Il est possible de forcer un index non géocentrique pour ce type de données, mais ce n'est pas conseillé par Oracle.

Pour gérer les coordonnées sphériques, il suffit d'associer aux données le SRID d'un système géocentrique, comme WGS84 (SRID=8307) par exemple.

La table `SDO_COORD_REF_SYS` liste tous les systèmes de projection disponibles ainsi que leur SRID.

Les calculs effectués sur les données géocentriques sont faits en mètres. Il est possible d'associer une précision, également en mètres, à un jeu de données. Les calculs seront alors faits en tenant compte de cette précision.

Oracle supporte l'expression des coordonnées géocentriques sous deux formes:

- avec des longitudes comprises entre 0° et 360°.
- avec des longitudes comprises entre -180° et +180°

Ainsi, les deux polygones suivants sont équivalents dans Oracle Spatial (avec un SRS géocentrique):

```
POLYGON ((350.0 -5.0, 10.0 -5.0, 10.0 5.0, 350.0 5.0, 350.0 -5.0))
POLYGON ((-10.0 -5.0, 10.0 -5.0, 10.0 5.0, -10.0 5.0, 10.0 -5.0))
```

Cependant, l'extension spatiale des données géocentriques, dans la table des métadonnées, doit être exprimée sous la forme: -180,+180,-90,+90.

Restrictions

Oracle impose un certain nombre de restrictions sur les systèmes géocentriques:

- Les cercles et arcs de cercles ne sont pas autorisés dans de tels systèmes. Il faut alors approximer les arcs de cercles en lignes, avec la fonction `SDO_GEOM.SDO_ARC_DENSIFY`;
- il n'est pas possible de stocker des polygones ayant une surface supérieure à la moitié de la surface de la terre;
- il n'est pas possible de stocker des lignes ayant des points espacés de plus de la moitié de la circonférence de la terre à l'équateur.

7.3.2. Gestion des SRS

Les SRS définis dans Oracle se basent principalement sur EPSG (cf. § sur la gestion des SRS dans PostGIS). C'est une nouveauté dans la version 10g.

Oracle associe également, comme spécifié par la norme OGC, un identifiant unique de SRS à toute donnée spatiale. Cet identifiant est le SRID de l'objet (Spatial Reference Identifier).

Oracle supporte comme SRID les codes EPSG et également ses propres codes issus des versions précédentes de Spatial.

Il existe des fonctions permettant de récupérer le code EPSG connaissant un SRID Oracle et inversement.

Oracle permet la création de nouveaux SRS si ceux fournis avec le système ne suffisent pas.

La création d'un nouveau SRS passe par l'insertion de ses caractéristiques (unités, projection, datum, etc.) dans les tables décrivant les SRS (tables du package *SDO_CS*), notamment la table *SDO_COORD_OPS* (pour les systèmes projetés) et *SDO_COORD_REF_SYSTEM* (pour les systèmes géocentriques).

8. Prédicats spatiaux

Sous ce terme de prédicats sont répertoriés des fonctions et/ou opérateurs permettant de tester les relations spatiales entre les objets.

Ces prédicats sont très couramment utilisés dans le monde des SIG car ils permettent de trouver des sous-ensembles d'objets ayant une relation spatiale particulière avec un ou plusieurs autres objets. Dans le cadre de Pleiades, par exemple :

- L'ensemble des segments intersectant la France,
- Tous les pays ne contenant aucun segment,
- Les segments a cheval sur plusieurs pays,
- etc.

Ces prédicats utilisent souvent la notion de rectangles englobant (bbox) des objets pour réaliser les tests, car il est plus rapide de faire des calculs sur des rectangles que sur des objets complexes, et les bbox servent de base aux index spatiaux éventuellement définis sur les colonnes. Les tests sont d'abord faits sur ces bbox avant de les faire sur les objets eux-mêmes, si ce deuxième calcul est pertinent.

La norme OGC définit un certain nombre de ces prédicats, qui doivent renvoyer une valeur booléenne ou une valeur évaluable dans une condition booléenne :

- Equals,
- Disjoint,
- Intersects
- Overlaps,
- Contains,
- Crosses,
- Within
- Touches,
- Relate,

Il faut noter que la norme définit très précisément chacun de ces prédicats en terme de relations entre les intérieurs, extérieurs et frontières des objets concernés. Il est important de se baser sur ces définitions lorsqu'on utilise ces prédicats, pour éviter une interprétation erronée des résultats.

Par exemple, la fonction touches() retourne faux pour deux polygones s'intersectant, comme présenté sur les schémas suivants :

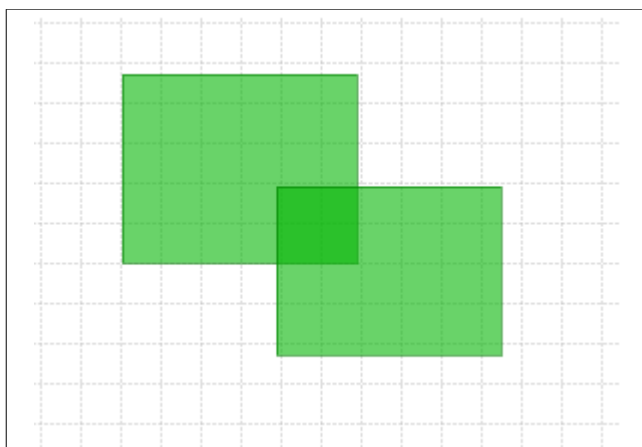


Illustration 7: Touches() renvoie faux pour ces deux polygones

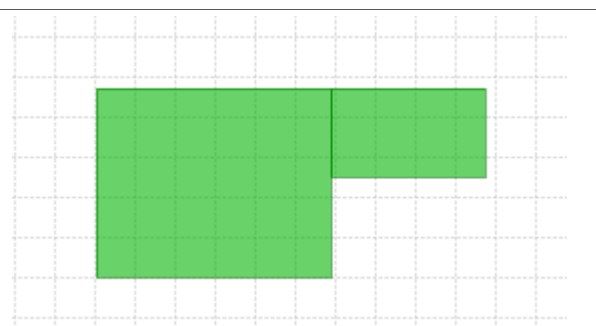


Illustration 8: Touches renvoie vrai pour ces polygones

8.1. Notes sur la fonction Relate

La fonction Relate définie par la norme OGC permet de tester les relations spatiales entre les intérieurs, les extérieurs et les frontières de deux objets géographiques.

Cette fonction prend en argument une valeur textuelle représentant la matrice des relations à tester pour les deux objets, sous la forme d'un texte exprimant les relations entre les deux objets.

Le schéma suivant montre cette matrice de relations (ou dim()) représente une fonction donnant la dimension maximum d'un objet: -1, 0, 1, 2, avec :

- -1 = l'ensemble vide,
- 0 = dimension d'un point,
- 1 = dimension d'une ligne,
- 2 = dimension d'un polygone :

	Intérieur (I)	Frontière (B)	Extérieur (E)
Intérieur (I)	$\dim(I(a) \cap I(b))$	$\dim(I(a) \cap B(b))$	$\dim(I(a) \cap E(b))$
Frontière (B)	$\dim(B(a) \cap I(b))$	$\dim(B(a) \cap B(b))$	$\dim(B(a) \cap E(b))$
Extérieur (E)	$\dim(E(a) \cap I(b))$	$\dim(E(a) \cap B(b))$	$\dim(E(a) \cap E(b))$

Table 2: Matrice des relations spatiales

La représentation de cette matrice sous la forme d'une chaîne de caractères se fait alors grâce à 9 caractères représentant les valeurs possibles de la matrice, (en la parcourant par lignes). Ces valeurs sont: T, F, *, 0, 1, 2, avec:

- T : True: toute valeur de dimension autre que l'ensemble vide (0, 1 ou 2)

- F : False: l'ensemble vide
- * : toute valeur: pas pris en compte
- 0 : La dimension de la relation doit être 0
- 1 : La dimension de la relation doit être 1
- 2 : La dimension de la relation doit être 2

Par exemple, pour exprimer le recouvrement de deux surfaces, on peut utiliser la fonction relate avec comme argument.

```
relate (geom1, geom2, 'T*T***T**')
```

Les autres prédicats sont des cas particuliers de la fonction Relate, plus explicites et plus faciles à utiliser.

8.2.MySQL

MySQL supporte les prédicats définis par la norme OGC avec la restriction importante que **ces fonctions n'agissent que sur les rectangles englobant des objets (bbox) et non sur les objets eux-mêmes.**

Il n'est pas indiqué la version à partir de laquelle ces fonctions agiront sur les objets.

8.3.PostGIS

PostGIS supporte tous les prédicats définis par la norme, en respectant le nommage.

Ces fonctions renvoient une valeur booléenne et s'utilisent de la façon suivante, par exemple pour trouver tous les identifiants de la table1 dont les objets recouvrent les objets de la table table2 :

```
select t1.gid from table1 t1, table2 t2 where t2.geom && t1.the_geom and overlaps(t1.the_geom, t2.the_geom);
```

(L'opérateur && est utilisé pour forcer l'usage des index spatiaux, si ceux-ci ont été définis sur les tables table1 et table2).

Lors des requêtes de test, ces opérateurs ont donné des résultats corrects. Tous les opérateurs spatiaux n'ont pas été testés dans toutes les combinaisons possibles. La certification à la norme OGC prouve que le système gère correctement tous les cas.

8.4.Oracle

Oracle Spatial et Locator supportent en partie les prédicats définis par la norme OGC. Tous les prédicats de la norme ne sont pas présents et les noms des prédicats Oracle ne correspondent pas à ceux de la norme.

Ceci peut poser un problème dans l'optique de l'écriture des requêtes spatiales génériques en se basant sur la norme OGC.

Oracle dispose de prédicats supplémentaires par rapport à la norme.

Au total, Oracle permet de qualifier toutes les relations spatiales entre objets, notamment par le support de la fonction Relate (*SDO_RELATE*).

Le tableau suivant liste les opérateurs spatiaux définis par la norme OGC et leur correspondance dans Oracle :

Opérateurs définis par la norme	Correspondance dans Oracle
Equals	SDO_EQUAL
Disjoint	La négation de SDO_ANYINTERACT peut être utilisé pour Disjoint : disjoint(a, b) = SDO_ANYINTERACT(a, b) <> 'TRUE';
Intersects	Tel que défini par la norme OGC, intersects(g1, g2) = Not(disjoint(g1, g2)). La fonction SDO_ANYINTERACT est donc équivalente à Intersects
Overlaps	SDO_OVERLAPS et également les versions SDO_OVERLAPBDYDISJOINT et SDO_OVERLAPBDYINTERSECT
Contains	SDO_CONTAINS
Crosses	Il n'y a pas d'équivalent dans Oracle. Il faut utiliser la fonction SDO_RELATE avec les bons paramètres ('T*T*****' ou 'O*****' suivant les cas), ou la fonction SDO_OVERLAPBDYDISJOINT dans le cas ligne-polygon
Within	SDO_INSIDE
Touches	SDO_TOUCH
Relate	SDO_RELATE

Table 3: Correspondance OGC-Oracle des opérateurs

Le tableau suivant liste les autres opérateurs Oracle:

Nom de l'opérateur	Usage
SDO_COVEREDBY	Pour tester si un objet est couvert par un autre
SDO_COVERS	Pour tester si un objet en couvre un autre
SDO_FILTER	Pour filtrer les objets qui peuvent avec une relation avec un objet donné. Cette fonction utilise les bbox des objets uniquement et utilise les index spatiaux définis sur les objets.
SDO_NN	Fonction pour la recherche du plus proche voisin
SDO_NN_DISTANCE	Fonction pour connaître la distance de l'objet renvoyé par SDO_NN. Ne peut être appelée que lors d'un appel à SDO_NN, ex: <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>SELECT c.mkt_id, c.name, SDO_NN_DISTANCE(1) dist FROM cola_markets c WHERE SDO_NN(c.shape, sdo_geometry(2001, NULL, sdo_point_type(10,7,NULL), NULL, NULL), 'sdo_num_res=2', 1) =</pre> </div>

	<code>'TRUE' ORDER BY dist;</code>
SDO_ON	Pour tester si un objet est « sur » un autre (l'intérieur et la frontière d'un objet se trouvent sur la frontière d'un autre objet, par exemple une ligne couvrant le contour d'un polygone)
SDO_WITHINDISTANCE	Pour trouver les objets situés a une certaine distance d'un objet.

Lors des requêtes de test, ces opérateurs ont donné des résultats corrects. (tous les opérateurs spatiaux n'ont pas été testés dans toutes les combinaisons possibles. La certification à la norme OGC prouve que le systèmes gère correctement tous les cas)

9.Fonctions spatiales

Les fonctions spatiales sont toutes les fonctions autres que les prédicats agissant sur les données spatiales.

Ces fonctions peuvent créer de nouveaux objets (*buffer, union, difference*), ou renvoyer des informations sur les objets : *geometryType, distance, area, srid, etc.*

Les fonctions créant de nouveaux objets sont les plus complexes et mettent en jeu des calculs importants.

C'est notamment le cas pour les opérateurs spatiaux : *union, intersection, difference, symmetric_difference*.

Les requêtes les mettant en œuvre sont souvent longues et il est très important d'indexer correctement les colonnes contenant les objets spatiaux pour garantir les meilleures performances.

Ces fonctions ont été regroupées dans les chapitres suivants:

- Fonctions d'information, pour les fonctions renvoyant des informations sur les objets spatiaux (dimensions, nombre de points ou de parties, etc.)
- Fonctions de création, pour les fonctions créant de nouveaux objets spatiaux a partir d'objets existant (*Buffer(), ConvexHull(), Intersection(), etc.*). Le chapitre 12.4.1 présente de façon graphique une partie des résultats de ces fonctions.

9.1.Fonctions d'information

La norme OGC définit un certain nombre de fonctions agissant sur les objets spatiaux (tous ou certains types uniquement).

Les documents FICHE1, FICHE2, FICHE3 et FICHE4 listent ces fonctions pour chaque système et indiquent le niveau de compatibilité avec la norme.

MySQL supporte la plupart des fonctions définies par la norme OGC concernant les informations sur les objets.

Ainsi, les fonctions *Envelope, Distance, Area, SRID, NumPoints, etc.* sont présentes.

Les fonctions *point_on_surface, isSimple* ne sont pas disponibles.

PostGIS et Oracle supportent l'ensemble des fonctions définies par la norme et en ajoute de

nouvelles, comme présenté au § Fonctions diverses.

9.2. Fonctions de création, Opérateurs spatiaux

9.2.1. MySQL

MySQL ne dispose d'aucune fonction de création d'objets spatiaux ou d'opérateurs spatiaux (tels que définis dans la norme OGC ou autres). Il n'est pas possible par exemple d'unir deux polygones ou de créer un tampon autour d'un point.

Cet limitation de MySQL est pénalisante lorsqu'on veut mettre en place un système de gestion de données spatiales performant et évolutif.

Les prochaines versions de MySQL devraient combler ce retard.

9.2.2. PostGIS

PostGIS propose toutes les fonctions de création et tous les opérateurs spatiaux définis par la norme OGC et en propose d'autres, enrichissant grandement les possibilités du système.

Les tests effectués dans le cadre de cette étude ont principalement portés sur les fonctions et opérateurs de la norme OGC.

Ces tests sont présentés dans le § Résultats des tests SQL.

Le document FICHE2 liste les fonctions définies par la norme et indique le support PostGIS pour ces fonctions.

Le § Fonctions Diverses décrit plus en détail les fonctions propres à PostGIS.

9.2.3. Oracle

Oracle Locator ne propose aucune fonction de création spatiale. Ces fonctionnalités sont ajoutées par la cartouche payante Spatial.

Oracle spatial supporte toutes les fonctions de création et tous les opérateurs spatiaux définis par la norme OGC et en propose d'autres, enrichissant grandement les possibilités du système.

Les tests effectués dans le cadre de cette étude ont principalement portés sur les fonctions et opérateurs de la norme OGC.

Les documents FICHE3 et FICHE4 listent les fonctions définies par la norme et indique le support Locator et Spatial pour ces fonctions.

Le § Fonctions Diverses décrit plus en détail les fonctions propres à Oracle Spatial

10. Import/Export

Avant de manipuler des objets spatiaux avec une base de données spatiale, il faut pouvoir charger des données dans cette base.

De même, après manipulation et transformation éventuelles des données, il peut être intéressant dans le cadre des SIG d'exporter les données de la base dans un format autre que celui de la base.

Les paragraphes suivant décrivent les moyens ou outils permettant de charger des données spatiales dans les différents systèmes :

10.1. MySQL

MySQL ne dispose pas d'outils pour importer les données depuis un autre format SIG. Pour insérer des données il faut utiliser la représentation textuelle (WKT) ou binaire (WKB) des objets spatiaux telles que définies par la norme OGC dans des requêtes SQL.

Ceci limite la facilité d'utilisation de MySQL, obligeant à une étape intermédiaire pour transformer des données spatiales ou à utiliser un programme externe (il existe peu de programmes externes pour réaliser cela. Certains d'entre eux sont présentés au chapitre 13.1).

10.2. PostGIS

PostGIS met à disposition un programme d'import des données et d'export des données spatiales.

Le programme shp2pgsql permet de charger des données au format ESRI Shapefile dans une base PostGIS, en créant les index et en mettant à jour les métadonnées.

Il permet d'associer un SRS aux données (SRID des objets).

Il génère un script SQL créant la table, ajoutant la colonne géométrique et chargeant les données dans la table.

Exemple d'utilisation :

```
shp2pgsql -s 27582 countries.shp countries
Cette commande peut être groupée avec une commande psql pour charger les
objets directement:
shp2pgsql -s 27582 countries.shp countries | psql -d mydb
```

Les paramètres du programme peuvent être affichés en exécutant le programme sans paramètres :

```
OPTIONS:
-s <srid> Set the SRID field. If not specified it defaults to -1.
(-d|a|c|p) These are mutually exclusive options:
-d Drops the table, then recreates it and populates
  it with current shape file data.
-a Appends shape file into current table, must be
  exactly the same table schema.
-c Creates a new table and populates it, this is the
  default if you do not specify any options.
-p Prepare mode, only creates the table.
-g <geometry_column> Specify the name of the geometry column
  (mostly useful in append mode).
-D Use postgresql dump format (defaults to sql insert statments.
-k Keep postgresql identifiers case.
-i Use int4 type for all integer dbf fields.
-I Create a GiST index on the geometry column.
-S Generate simple geometries instead of MULTI geometries.
-w Use wkt format (for postgis-0.x support - drops M - drifts
```

```
coordinates).
-W <encoding> Specify the character encoding of Shape's
  attribute column. (default : "ASCII")
-N <policy> Specify NULL geometries handling policy (insert,skip,abort)
-? Display this help screen
```

Le programme `pgsql2shp` permet de créer des fichiers au format ESRI Shapefile à partir d'une table spatiale. (Exportation des données) par exemple :

```
pgsql2shp -f /tmp/countries.shp test_spatial countries
```

Les paramètres du programme peuvent être affichés en exécutant le programme sans paramètre :

```
RCSID: $Id: pgsq2shp.c 2513 2006-10-14 14:22:10Z mschaber $ RELEASE: 1.1.6
USAGE: pgsq2shp [<options>] <database> [<schema>.]<table>
       pgsq2shp [<options>] <database> <query>

OPTIONS:
-f <filename> Use this option to specify the name of the file
  to create.
-h <host> Allows you to specify connection to a database on a
  machine other than the default.
-p <port> Allows you to specify a database port other than the default.
-P <password> Connect to the database with the specified password.
-u <user> Connect to the database as the specified user.
-g <geometry_column> Specify the geometry column to be exported.
-b Use a binary cursor.
-r Raw mode. Do not assume table has been created by
  the loader. This would not unescape attribute names
  and will not skip the 'gid' attribute.
-k Keep postgresql identifiers case.
-? Display this help screen.
```

10.3. Oracle

Oracle met à disposition les sources d'un programme Java permettant de charger des données au format ESRI Shapefile dans une base Oracle.

Avant d'utiliser le programme, il est nécessaire de le compiler avec le compilateur Java fourni avec Oracle.

Il peut ensuite s'utiliser en ligne de commande, de la façon suivante :

```
/var/oracle/app/oracle/product/10.2.0/db_1/jdk/bin/java -cp
.:$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$ORACLE_HOME/md/lib/sdoutl.jar:$ORACLE_
HOME/md/lib/sdoapi.jar shp2sdo -h localhost -p 1521 -s testspatial -u nic
-d cnes -t COUNTRIES -f data/cntry98 -g the_geom -i gid -r 8307 -x
'-180,180' -y '-90,90' -a
```

Note: dans cette étude, ce programme a été compilé sous le nom « `shp2sdo` ».

Les paramètres du programme peuvent être affichés en exécutant le programme sans paramètre :

```
USAGE: java -cp
[ORACLE_HOME]/jdbc/lib/ojdbc14.jar;./sdoutl.jar;./sdoapi.jar shp2sdo -h
db_host -p db_port -s db_sid -u db_username -d db_password -t db_table -f
shapefile_name [-i table_id_column_name][-r srid][-g db_geometry_column][-x
max_x,min_x][-y max_y,min_y][-o tolerance]
Usage explanation (parameters used):
<-h>: Host machine with existing Oracle database
<-p>: Host machine's port with existing Oracle database (e.g. 1521)
<-s>: Host machine's SID with existing Oracle database
<-u>: Database user
<-d>: Database user's password
<-t>: Table name for the result
<-f>: File name of an input Shapefile (without extension)
[-i]: Column name for unique numeric ID; if required
[-r]: Valid Oracle SRID for coordinate system; use 0 if unknown
[-g]: Preferred or valid SDO_GEOMETRY column name
[-x]: Bounds for the X dimension; use -180,180 if unknown
[-y]: Bounds for the Y dimension; use -90,90 if unknown
[-o]: Load tolerance fields (x and y) in metadata, if not specified,
tolerance fields are 0.05
[-a]: Append shapefile data to an existing table
[-n]: Start ID for column specified in -i parameter
[-c]: Commit interval. Default, only commits at the end of a run.
```

11. Fonctions diverses

Oracle Spatial et PostGIS dans une moindre mesure, proposent une série de fonctions agissant sur les objets spatiaux, en plus de celles définies par la norme OGC.

MySQL ne propose pas de fonctions en plus de celles définies par la norme OGC, partiellement implémentées.

11.1. PostGIS

PostGIS propose un certains nombres de fonctions pour manipuler les objets spatiaux, en plus de celles définies par la norme OGC.

Parmi ces fonctions, on trouve:

- Des fonctions pour gérer les tables spatiales :
 - ① DropGeometryTable : suppression d'une table spatiale et de ses métadonnées
 - ① update_geometry_stats : mise a jour des statistiques de la table,
- Des opérateurs spatiaux agissant sur les bbox des objets :
 - ① && : opérateur intersects
 - ① @ : opérateur contained by
 - ① ~= : opérateur same as

- ① etc...
- Des fonctions de mesure, comme :
 - ① distance_sphere, permettant de calculer la distance de deux points sur une sphère
 - ① distance_sphéroid, permettant de calculer la distance de deux points sur un sphéroïde
 - ① length3d, pour mesurer la longueur d'une ligne en 3 dimensions
 - ① length_sphéroid, pour mesurer la longueur d'une géométrie sur un sphéroïde
 - ① max_distance, pour mesurer la distance maximum entre deux lignes
 - ① perimeter, pour calculer le périmètre d'un polygone ou multipolygone
 - ① azimuth, pour trouver l'azimuth du segment formé par les points passés en paramètres
- Des fonctions de formatage des données spatiales :
 - ① AsEWKT, pour écrire un objet sous forme de WKT étendu (format propre à PostGIS, incluant le SRID de l'objet).
 - ① AsSVG, pour écrire un objet sous la forme d'un chemin SVG (Scalable Vector Graphics, <http://www.w3.org/Graphics/SVG/>)
 - ① AsGML, pour écrire un objet sous la forme d'un élément GML (Geographic Markup Language, <http://www.opengis.net/gml/>), qui est la norme d'échange de données vectorielles OGC
 - ① AsKML, pour écrire un objet sous forme de lien Google Map (à partir de la version 1.2 uniquement)
- De nombreuses fonctions de construction d'objets, à partir de texte, de nombres ou d'autres objets :
 - ① MakePoint, pour créer un point à partir de deux ou trois nombres
 - ① MakeBox2D, pour créer une bbox avec deux points
 - ① MakeLine, pour créer une ligne à partir d'un ensemble de points (on peut classer les points dans une sous requête avant d'utiliser cette fonctionnalité)
 - ① MakePolygon, pour créer un polygone à partir d'un ensemble de lignes fermées (extérieur et intérieurs)
 - ① BuildArea, pour construire un polygone
 - ① Polygonize, pour construire des polygones constitués par le réseau passé en paramètres.
 - ① Collect, pour regrouper une série d'objets dans une collection. Fonction d'agregation, comme sum() ou max()
 - ① Dump, pour extraire des collections en une série d'objets.
- Des fonctions d'édition d'objets, permettant de modifier ou compléter un objet:
 - ① AddBBox, pour ajouter une bbox a un objet et ainsi augmenter la performance des requêtes agissant sur les bbox (DropBBox pour supprimer)
 - ① AddPoint, pour ajouter un point a une ligne, (removePoint pour supprimer un point)

- ① SetPoint, pour remplacer un point a une certaine position
- ① Multi, pour forcer le type de l'objet en multi objet.
- ① Transform, pour changer le SRS d'un objet (voir le § sur les systèmes de référence spatiale)
- ① Affine, pour appliquer une transformation affine 2D ou 3D a un objet
- ① Translate, pour translater un objet
- ① Scale, pour mettre à l'echelle un objet en multipliant ses coordonnées par un paramètre.
- ① RotateX, RotateY, RotateZ, pour faire tourner un objet suivant les axes X, Y, ou Z
- ① Reverse, pour retourner l'ordre des points dans un objet.
- ① ForceRHR, pour forcer le sens de parcours des polygones dans le sens trigonométriques (les trous tournant dans le sens horaire)
- ① Simplify, pour supprimer des points dans un objet tout en conservant sa forme (algorithme Douglas Peuker)
- ① SnapToGrip, pour accoler les points d'un objet a une grille virtuelle (permet de fixer la précision des données).
- Des fonctions de référencement linéaire, pour gérer les données de type ligne portant des informations spécifiques sur chaque segment (lignes mesurées):
 - ① line_interpolate_point, pour interpoler une point sur une ligne
 - ① line_substring, pour extraire une partie d'une ligne, en précisant une certaine fraction de la ligne de départ
 - ① locate_along_measure, locate_between_measures, pour extraire un objet à partir d'une mesure
 - ① etc...
- Des fonctions diverses, comme:
 - ① summary, pour avoir un résumé textuel d'un objet spatial
 - ① box2d, box3d, pour retourner la bbox d'un objet
 - ① extent, pour connaître la bbox d'une colonne spatiale (fonction d'agrégation)
 - ① nrings, pour connaître le nombre de parties d'un polygone
 - ① npoints, pour connaître le nombre de points formant un objet
 - ① isValid, pour tester la validité (telle que définie par la norme OGC) d'un objet
 - ① expand, pour retourner une bbox étendue d'une certaine valeur
 - ① find_srid, pour chercher dans les métadonnées le SRID d'une colonne spatiale dont on connait le nom.
 - ① xmin, ymin, zmin, xmax, ymax, zmax pour trouver les points extrêmes d'une bbox d'un objet.
 - ① etc.
- Des fonctions respectant la norme SQL-MM. Ces fonctions sont apparues à la version 1.2

de PostGIS et n'ont donc pas été testés dans le cadre de cette étude.

11.2. Oracle Spatial

Oracle spatial offre un très grand nombre de fonctions spatiales en plus de celles définies par la norme OGC.

Oracle regroupe ces fonctions en thèmes ou package et couvrent l'ensemble des fonctionnalités d'un SIG, notamment:

- Des fonctions permettant le géocodage de données, c'est à dire la possibilité de créer des points à partir d'adresses postales.
- Des fonctions de gestion des référencements linéaires (lignes permettant de manipuler les réseaux, routiers, par exemples)
- des fonctions de gestion des images (raster) en base, avec opérations de calcul, stockage sous forme compressée, etc.
- des fonctions pour mesurer la performance des requêtes spatiales ou donner des informations sur les objets spatiaux (taille de l'index sur la colonne, bbox moyenne pour une colonne)
- Des fonctions d'analyse et de recherche d'information (Data Mining)

Chacun de ces packages contient de nombreuses fonctions et il serait fastidieux de les lister ici. Pour plus d'information, se référer à la document Oracle (cf. Annexe)

12. Requêtes SQL de test

Cette étude a pour but de comparer les caractéristiques de 3 systèmes spatiaux vis a vis d'une norme et des cas d'utilisation du projet Pleiades.

Un certain nombre de requêtes spatiales ont été exécutées sur chacun des systèmes (pour autant que le système supportait la fonction à tester dans la requête) dans le but de vérifier le comportement du système (temps de réponse, exactitude de la réponse, etc.)

Le paragraphe 4, « Conditions de l'étude » indique les outils utilisés pour ces tests SQL.

Les requêtes ont été regroupées en plusieurs catégories :

1. Les requêtes donnant des informations sur les objets spatiaux: *area*, *length*, *dimension*, *srid*, *geometryType*, etc. (syntaxe propre à chaque système)
2. Les prédicats spatiaux manipulant les rectangles englobant des objets (bbox)
3. Les prédicats spatiaux manipulant les objets eux-mêmes : *within*, *intersects*, *contains*, *overlaps*, etc.
4. Les opérateurs spatiaux créant de nouveaux objets : *union*, *intersection*, *buffer*, etc.

Les résultats des requêtes des catégories 1 à 3 sont présentés dans la fiche de synthèse des requêtes SQL DOC_SQL

Les résultats des requêtes SQL concernant la création d'objets par des opérateurs spatiaux sont présentés sous forme graphique dans un sous-chapitre. Ces fonctions ne sont pas disponibles avec MySQL et ne sont donc pas présentées dans le tableau de synthèse.

12.1. Requêtes d'information

Les 3 systèmes (Oracle Spatial et Locator se comportant de la même façon) donnent des résultats corrects pour les fonctions testées dans cette catégories.

Les mesures de distance ou de surface ne sont pas comparables entre elles car Oracle réalise ses calculs dans un espace sphérique respectant la forme de la terre correspondant au SRS des données, alors que MySQL et PostGIS travaille dans un espace plan. Les résultats entre MySQL et PostGIS sont comparables entre eux et produisent les mêmes résultats, à quelques décimales près. La différence de résultat porte sur la 8^{ème} ou 9^{ème} décimale, bien en deçà de la précision des mesures, quelque soit l'unité de mesure des données.

On constate que MySQL est très performant sur ce type de requêtes, avec des temps souvent inférieur a 50 millisecondes.

PostGIS et Oracle ont des temps d'exécution comparables, de l'ordre de quelques dizaines a quelques centaines de millisecondes.

Ces mesure sont a comparer entre elles, pas a comparer dans l'absolu.

12.2. Prédicats spatiaux sur les bbox

Ces fonctions permettent de tester les relations entre les objets en se basant sur leur bbox. Ces requêtes, tirant parti des index spatiaux, sont souvent performantes et servent de base aux autres requêtes, en éliminant beaucoup d'objets de la liste de résultat.

Les résultats sont généralement comparables entre les 3 systèmes.

A noter la requête concernant les segments intersectant des pays qui retourne moins de résultats dans Oracle, du au fait qu'Oracle effectue ses calculs dans un repère sphérique. Dans un repère plan, comme c'est le cas pour PostGIS et MySQL, les bbox sont très exagérées pour les latitudes élevées et basses (près des pôles). Les intersections entre objets sont alors plus nombreuses.

Ces différences de résultats n'apparaissent plus quand les objets impliqués sont petits par rapport à la surface de la terre. Les deux types de calculs donnant des résultats proche dans ce cas.

Là encore, MySQL exécute les requêtes plus rapidement que les deux autres systèmes pour la plupart des requêtes.

A noter également que la requêtes concernant le nombre de pays n'intersectant aucun segment prend plusieurs dizaines de minutes dans Oracle, contre 13 secondes dans PostGIS et 450ms dans MySQL.

12.2.1. Problème sur Oracle ?

Lors du test de l'opérateur agrégé *Union* réalisant l'union de plusieurs polygones (fonction de table), un résultat surprenant est apparu dans Oracle:

L'union de tous les pays de la table *CS* renvoie plus de résultats dans Oracle que dans PostGIS. Dans Oracle, le Danemark est inclus dans la liste des pays a unir.

Après vérification, il apparaît clairement que le polygone représentant la fenêtre d'intersection et le rectangle englobant (bbox) du Danemark ne s'intersectent pas, comme montré sur l'image

ci-après.

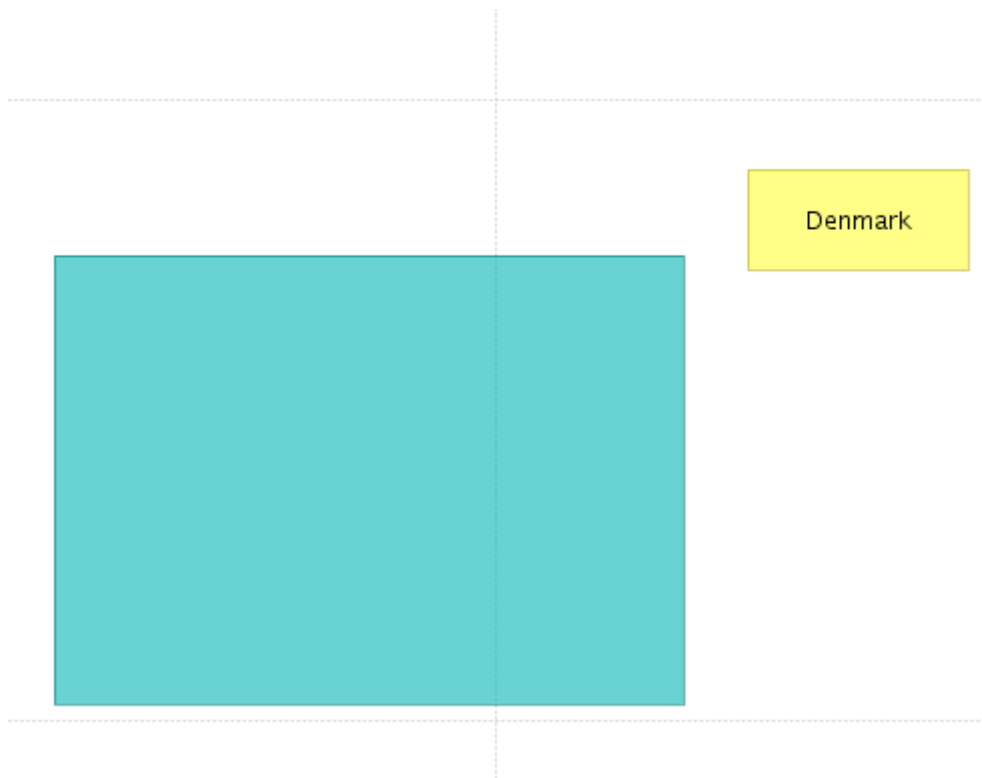


Illustration 9: bbox du Danemark (en jaune) et de la fenêtre de requête (en bleu).

L'opérateur *SDO_FILTER*, sensé agir sur les rectangles englobant (bbox) des objets, considère que les bbox s'intersecte.

L'utilisation des prédicats agissant sur les objets, comme *Relate()* et ses dérivés, donnent des résultats corrects.

Le comportement du prédicat *SDO_FILTERs* est donc sujet a caution et conduit en tout cas a un résultat faux, comme présenté graphiquement au § « 12.4.1 Résultats Graphiques ».

Il faudrait pousser plus avant les tests pour savoir comment se comporte exactement le prédicat *SDO_FILTER*.

12.3. Prédicats spatiaux sur les objets

MySQL ne supporte pas ce type de prédicats dans la version testée dans cette étude. Les prédicats sont définis mais appellent en fait les fonctions agissant sur les bbox. Les résultats de ces requêtes ne peuvent pas être comparés au 2 autres systèmes.

Les systèmes PostGIS et Oracle renvoient des résultats comparables, dans des temps d'exécution à peu près comparables, sauf pour certaines requêtes, dont les temps d'exécution sont présentés dans le document DOC_SQL.

A noter que certaines requêtes dans Oracle ont pris des temps d'exécution anormalement longs : plusieurs dizaines de minutes.

C'est par exemple le cas des prédicats lorsque la première géométrie passée en paramètre provenait de la table des pays.

La requête testant le prédicat « touches », entre les segments et les pays, prend 6 secondes dans le cas touches(segment, pays) et prend plus de 10 minutes dans le cas touches(pays, segment).

L'ordre des paramètres pour les prédicats spatiaux d'Oracle joue donc un rôle important dans la performance des requêtes.

Hormis ces requêtes, les temps d'exécution sont à peu près comparables dans les deux systèmes.

12.4. Opérateurs spatiaux

MySQL ne supporte pas de fonctions créant des objets, comme *union* ou *buffer*.

A l'exception notable de l'opérateur *SDO_FILTER* associé avec une union agrégée, les deux autres systèmes donnent des résultats comparables, comme présenté dans le chapitre suivant.

Le cas de du prédicat *SDO_FILTER*, filtre primaire agissant sur les bbox, est présenté au chapitre 12.2.1. Oracle semble prendre en compte des polygones plus grands que les bbox des objets, conduisant à des résultats faux quand ce filtre primaire est utilisé pour réaliser des opérations sur les objets (*union* par exemple).

Les temps d'exécution sont en général plus rapides dans PostGIS, sauf pour les fonctions *buffer* et *convexHull*, plus rapides dans Oracle.

A noter que la requête réalisant l'union de tous les pays prends plusieurs heures dans Oracle, contre 1min40s dans PostGIS.

Le résultat de la fonction *buffer* n'est pas le même entre les systèmes :

Oracle travaillant en coordonnées sphérique, la forme des tampons générés est différente de celle générée par PostGIS et MySQL qui travaillent en coordonnées planes. Les graphiques du chapitre suivant illustrent bien cela.

La fonction *convexHull* pour le pays Russie renvoie un polygone invalide dans Oracle, se recoupant lui-même. Ceci est du au fait que le polygone lui-même est invalide dans Oracle sûrement du au mauvais découpage du polygone aux longitudes extrêmes (180°).

Dans PostGIS le résultat est correct car le polygone suit la même règle de gestion des longitudes que PostGIS.

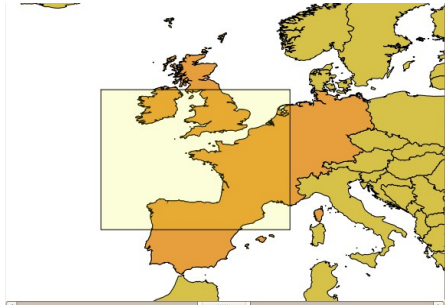
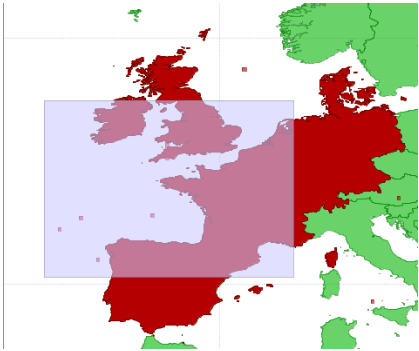
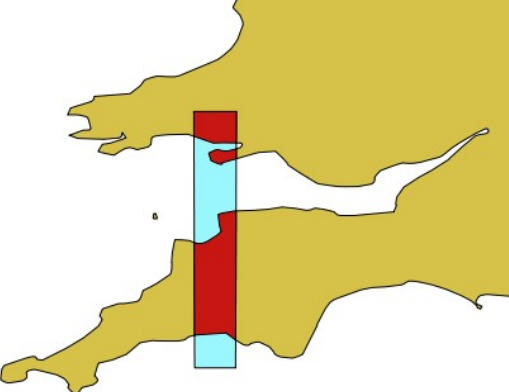
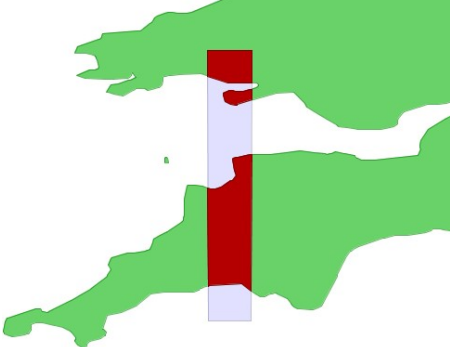


12.4.1. Résultats graphiques

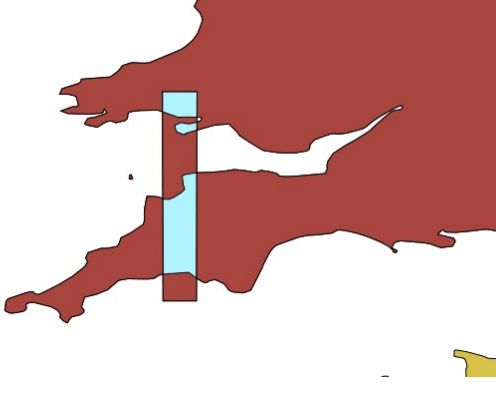
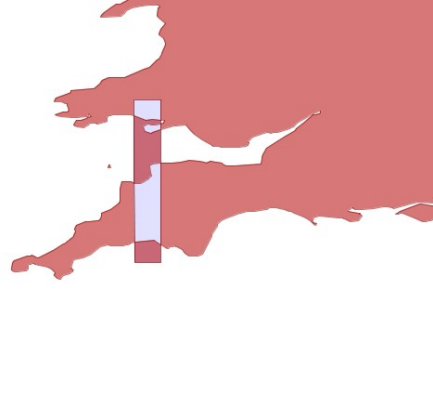
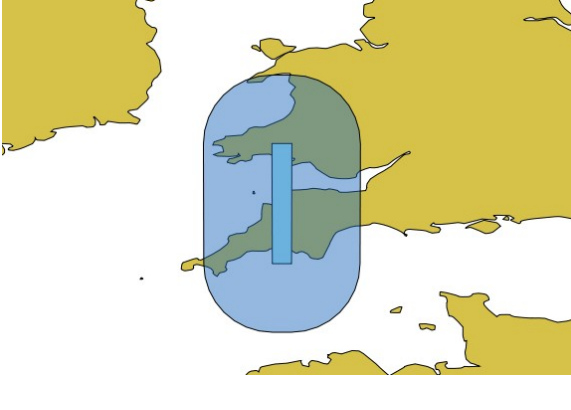
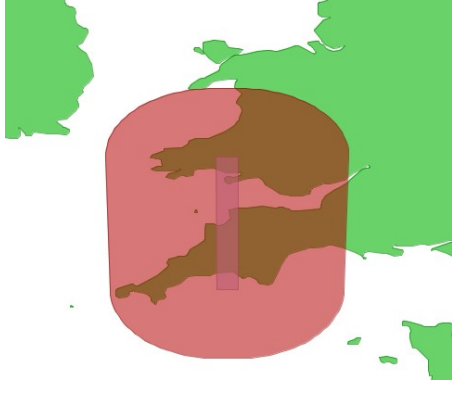
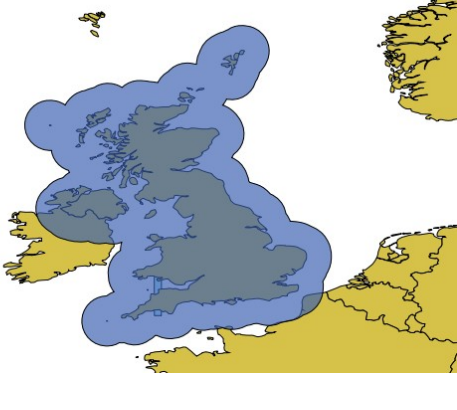
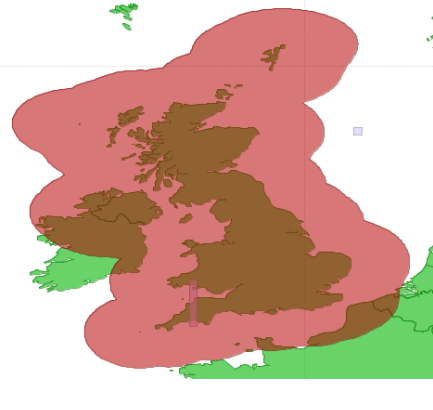
L'affichage des résultats sous forme graphique a été réalisé avec deux logiciels OpenSource gérant le format WKT : OpenJump (www.openjum.org) et MezoGIS (www.mezogis.org) et permettant d'exporter les cartes sous forme d'images PNG.

OpenJump est un SIG écrit en Java, gérant de façon native le format shapefile et PostGIS et permettant de visualiser des objets au format WKT.

MezoGIS est un outil de visualisation pour PostGIS, permettant d'afficher les objets géographiques issus de requêtes ou de tables enregistrées. Il permet de contrôler très rapidement les objets graphiques présents dans PostGIS.

Le tableau suivant présente de façon graphique le résultat des requêtes spatiales décrites dans le document `requetes_sql.pdf`

requêtes	PostGIS	Oracle Spatial
<p>Union agrégée de tous les pays intersectant un extent couvrant l'Europe de l'ouest</p>		
<p>L'extent est représenté par le rectangle en semi-transparence (jaune et bleu clair). Le résultat est présenté en orange et rouge, respectivement. On note que le résultat n'est pas le même entre PostGIS et Oracle: Oracle inclut le Danemark dans l'union des polygones, pas PostGIS. Le filtre primaire d'Oracle prend en compte le Danemark alors que visuellement, la bbox de ce pays n'intersecte pas l'extent couvrant l'Europe. (cf. § 12.2.1)</p>		
<p>Intersection du segment 929 avec les pays</p>		
<p>Intersection matérialisée en rouge</p>		
<p>Différence du segment 323 avec les pays</p>		

requêtes	PostGIS	Oracle Spatial
La partie extraite est le carré bleu au centre de l'image. Différence en rouge.		
Différence symétrique (XOR) du segment 929 avec les pays		
Seul le premier résultat est montré ici (XOR avec la grande bretagne). La différence symétrique est montrée en rouge, le segment en bleu.		
tampon de 1° ou 100 km autour du segment 929		
tampon de 1° sur l'image de gauche, de 100km sur l'image de droite. Oracle (à droite), réalisant les opérations en coordonnées métriques, la forme du tampon n'est pas la même que sur l'image de gauche.		
tampon de 1° ou 150 km autour du pays Grande Bretagne		
tampon de 1° (à gauche) ou 150 km (à droite). Oracle réalisant les opérations en coordonnées métriques, la forme du tampon est différente entre les deux images		

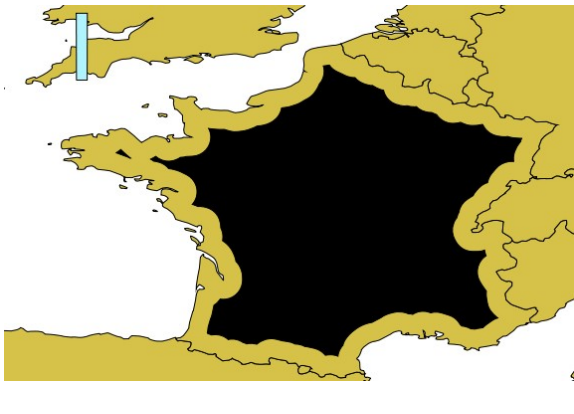
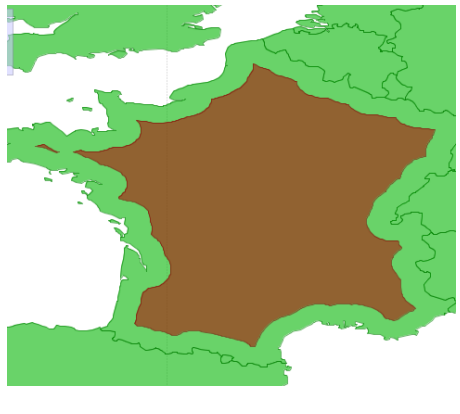
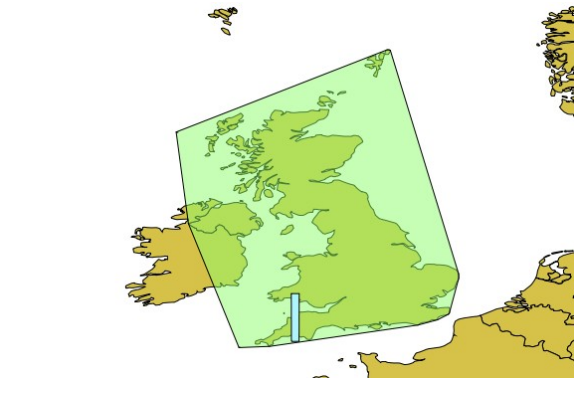

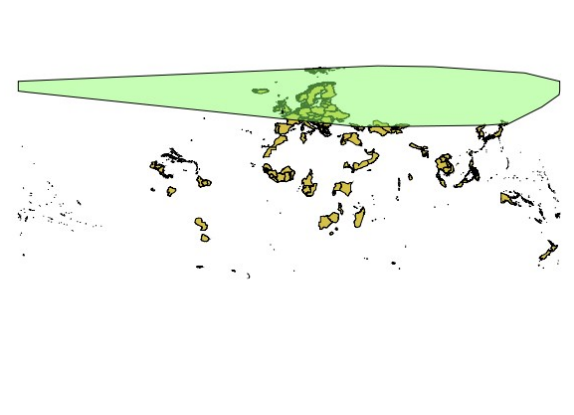
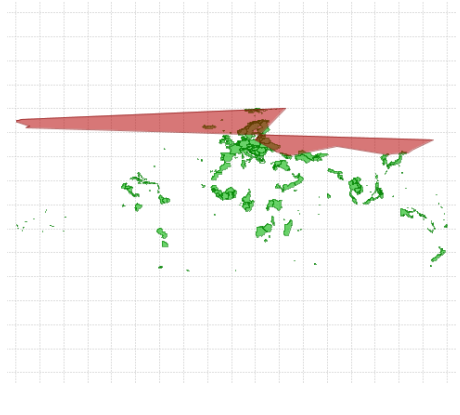
requêtes	PostGIS	Oracle Spatial
<p>tampon négatif de 0.5° ou 50 km (oracle) autour du pays France. à cetà cette latitude, 0.5° ~ 50 km</p>		
<p>tampon négatif de 0.5° ou ???</p>		
<p>Enveloppe convexe du pays Grande Bretagne</p>		
<p>la forme des polygones est la même.</p>		
<p>Enveloppe convexe du pays Russie</p>		
<p>Autointersection sur le polygone Oracle (pb des polygones couvrant toutes les latitudes.)</p>		

Table 4: Objets graphiques issus des requêtes spatiales

13. Paramètres de performance

Régler les paramètres d'un système de base de données pour optimiser ses performances est une opération délicate. C'est le rôle de l'administrateur de la base de données.

Suivant l'utilisation que l'on va faire du système, et suivant la configuration de la machine, les paramètres d'un systèmes ne seront pas les mêmes.

Les données géographiques stockées dans les systèmes présentés sont généralement volumineuses, tant en taille d'objets (polygones ou lignes contenant un grand nombre de points), qu'en nombre d'objets (table de plusieurs millions de points représentant des toponymes par exemple).

13.1. Remplissage des tables

Lors du remplissage de tables spatiales devant contenir beaucoup de données, il peut être intéressant de ne créer l'index spatial qu'après l'insertion. En effet, lors de chaque insertion, l'index spatial doit être mis à jour. Cette opération prend d'autant plus de temps qu'il y a d'objets dans la table.

Les étapes pour charger une table volumineuse sont alors:

- Définir la structure de la table;
- insérer les données (requêtes SQL, programme de chargement, etc.);
- créer l'index spatial sur la table;

Pour chacun des systèmes, quelques recommandations de paramétrage sont présentées ci-après.

13.2. MySQL

Les paramètres des systèmes doivent être changés dans le sens de l'optimisation de tables volumineuses, en privilégiant en premier la vitesse des disques, puis la taille de la mémoire vive, et enfin la puissance du processeur sur la machine hébergeant le système.

La cartouche spatiale étant récente sur MySQL, il n'y a pas encore de documentation pertinentes sur l'optimisation de la partie spatiale dans MySQL. L'effort d'optimisation doit porter sur l'accès aux tables contenant soit beaucoup de données, soit des données volumineuses, soit les deux.

Le chapitre 7 du guide utilisateur traite de l'optimisation de MySQL (<http://dev.mysql.com/doc/refman/5.1/en/optimization.html>)

13.3. PostGIS

Le stockage de données volumineuses dans des tables PostGIS peut être optimisé en utilisant les fonctionnalités de PostgreSQL.

Les premiers réglages de performance de PostgreSQL/PostGIS concernent les paramètres généraux de PostgreSQL, définis dans le fichier PG_HOME/main/postgresql.conf.

Il faut veiller à ce que les paramètres de mémoire, de taille des tampons, etc. correspondent aux capacités de la machine en terme de mémoire et de disques durs.

Il existe de nombreuses sources de références sur le net concernant ce sujet. Citons par exemple:

- La documentation PostgreSQL, chapitre 13: Performance Tips
- Tuning PostgreSQL for performance (<http://www.varlena.com/GeneralBits/Tidbits/perf.html>)
- Performance Tuning PostgreSQL (<http://www.revsys.com/writings/postgresql-performance.html>)
- PostgreSQL Performance Tuning | LinuxJournal (<http://www.linuxjournal.com/article/4791>)

La documentation PostGIS consacre également un chapitre au guide de performance. Les principaux éléments sont résumés dans les chapitres suivants.

Le gain de performance sur une base correctement configurée peut être très important, notamment en optimisant les mécanismes de cache qui évitent au système de recharger des données depuis le disque (opération la plus coûteuse dans la chaîne de traitement de l'information).

13.3.1. Données volumineuses

Pour les tables contenant des données très volumineuses, mais peu de données, l'optimisateur de requêtes PostgreSQL peut considérer dans certains cas qu'il est plus rentable de faire un parcourt séquentiel de la table au lieu d'utiliser un index spatial, qui serait plus performant dans ce cas, car les objets à tester sont volumineux.

Pour contourner ce problème, il est possible, avant de lancer la requête SQL, de forcer l'utilisation des index en exécutant la commande:

```
SET enable_seqscan TO OFF;
```

puis en revenant au mode par défaut après l'exécution de la requête:

```
SET enable_seqscan TO ON;
```

Un autre moyen de contourner cette mauvaise estimation est de créer une colonne dans la table en question contenant les bbox des objets, et d'indexer cette colonne. On utilisera alors l'opérateur && sur les objets de cette colonne lors des requetes, ex:

```
SELECT
addGeometryColumn('myschema','mytable','bbox','4326','GEOMETRY','2');
UPDATE mytable set bbox = Envelope(Force_2d(the_geom));

SELECT geom_column FROM mytable WHERE bbox &&
SetSRid('BOX3D(0 0,1 1)'::box3d,4326);
```

13.3.2. Réordonner les lignes de la table

Pour les tables contenant un grand nombre de données, qui sont accédées souvent en lecture et qui contiennent un index, il est possible de réordonner leurs lignes pour les faire correspondre aux critères de recherche de l'index.

C'est la commande CLUSTER qui permet de faire cela.

Le gain se situe a deux niveaux, d'abord en limitant les accès a la table, ensuite, si les données de travail ne représente qu'une petite portion de l'index, les mécanismes de cache sont plus efficaces, car les données ne sont pas reparties sur un grand nombre de pages physiques sur le disque.

Exemple:

```
CLUSTER my_geom_index ON my_table;
```

Note: il n'est pas possible d'utiliser la commande CLUSTER sur une colonne contenant des valeurs nulles. On peut modifier le type de la colonne avec la commande:

```
ALTER TABLE my_table ALTER COLUMN the_geom SET not null;
```

13.4.Oracle

L'optimisation d'Oracle est une tâche délicate, car de nombreuses configuration de travail et paramètres de réglages sont disponibles.

Ce sujet est l'objet d'une documentation propre mis a disposition par Oracle:

Oracle® Database Performance Tuning Guide (http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14211/toc.htm), ainsi que de plusieurs forums d'utilisateurs (par exemple: <http://www.oraFAQ.com/forum/f/6/0/>)

Oracle propose un certain nombre d'outils pour analyser, identifier et corriger des problèmes de performance sur un système opérationnel.

13.4.1.Package SDO_TUNE

Concernant la partie spatiale, Oracle dispose d'un package spécial appelé SDO_TUNE contenant des fonctions permettant par exemple de calculer la taille d'un index spatial sur une table pour optimisation éventuelle de l'espace de stockage de cet index, ou encore de tester la qualité d'un ou plusieurs index spatiaux en retournant une valeur de dégradation de la qualité des index. Cette valeur est un nombre indiquant combien de temps en plus une requête spatiale prendrait par rapport a la même requête faite sur des index mis a jour.

Un chiffre élevé, de 2 ou 3, indique que l'index courant prend 2 ou 3 fois plus de temps à parcourir que lors de sa création ou de sa dernière mise a jour. Signe qu'il faut reconstruire l'index spatial sur la table avec la commande:

```
ALTER INDEX [schema.]index REBUILD
```

14.Portage des requêtes spatiales

La norme OGC sur laquelle s'appuient ces 3 systèmes pour gérer les données géographiques définit un modèle d'objets et de fonctions agissant sur ces objets. Cette norme fait appel aux fonctionnalités objets des systèmes de base de données, permettant d'ajouter de nouveaux type et de nouvelle fonctions.

Le respect des noms des types et des fonctions tels que définis par la norme est la première

étape pour assurer le portage de requêtes spatiales entre les 3 systèmes.

Malheureusement, Oracle n'adhère pas aux noms définis dans la norme et utilise ses propres noms pour les fonctions spatiales (par exemple : *SDO_BUFFER* pour *buffer*, *SDO_GEOM.SDO_INTERSECTION* pour *intersection*, etc).

De plus, chaque système apporte ses propres contraintes dans l'écriture des requêtes spatiales:

- PostGIS impose l'utilisation d'un opérateur spatial (&&, @, ||, etc.) entre deux objets spatiaux pour garantir l'utilisation d'éventuels index spatiaux,
- Oracle renvoie une valeur textuelle, 'TRUE' ou 'FALSE' pour les prédicats spatiaux, alors que MySQL et PostGIS renvoient une valeur booléenne directement évaluable. La norme OGC précise que les prédicats doivent renvoyer une valeur entière: 1 pour vrai, 0 pour faux, -1 pour inconnu, quand le prédicat est invoqué avec des arguments nuls
- MySQL appelle la fonction *length* de la norme OGC : *glength*, car *length* est déjà le nom d'une fonction.
- PostGIS appelle la fonction « *union* » de la norme: *geomunion*, car *union* est un mot réservé en SQL,
- etc.

L'analyse des fonctions définies par chaque système montre qu'un portage partiel des requêtes spatiales entre PostGIS et MySQL est possible, lorsque ces requêtes ne mettent pas en œuvre des index spatiaux (obligation d'utiliser des opérateurs spécifiques dans ce cas dans PostGIS).

On peut envisager différents moyens pour assurer un portage plus global des requêtes spatiales dans les 3 systèmes. Les paragraphes suivants en indiquent deux.

14.1. Créer une surcouche applicative

Pour s'assurer du portage de requêtes spatiales entre les 3 systèmes, une piste serait de développer une surcouche définissant les objets et fonctions spatiales pour chaque système, de sorte à pouvoir réaliser une correspondance entre les systèmes. Il faudrait également inclure dans cette surcouche des règles de construction de requêtes propres à chaque système (par exemple pour utiliser les index spatiaux dans tous les cas.

On pourrait par exemple envisager d'écrire les requêtes en pur SQL 92, en se conformant strictement à la norme pour les noms des types et des fonctions, puis traduire cette requêtes dans chaque système grâce à l'outil développé.

Par exemple, pour trouver les identifiants des segments intersectant des pays:

La requête générique pourrait être :

```
select count(d.id)
from datastrip d, cs c
where intersects(d.contour, c.the_geom) = 1;
```

La requête MySQL serait alors :

```
select count(d.id)
from datastrip d, cs c
where intersects(d.contour, c.the_geom);
```

La requête PostGIS :

```
select count(d.id) from datastrip d, cs c where d.contour && c.the_geom  
and intersects(d.contour, c.the_geom);
```

La requête Oracle :

```
select count(d.id)  
from datastrip d, cs c  
where sdo_anyinteract(d.contour, c.the_geom) = 'TRUE';
```

On le voit, s'assurer du portage des requêtes spatiales entre les 3 systèmes est une tâche difficile si elle doit être faite de façon automatique, car il faut prendre en compte les spécificités et limitations des 3 systèmes.

L'adhésion des systèmes à la norme OGC (partielle pour MySQL) ne garantit donc pas une compatibilité complète des requêtes spatiales.

14.2. Utiliser un outil de correspondance Objet/Relationnel

Ces outils définissent un niveau d'abstraction entre un modèle relationnel (base de données par exemple) et un modèle objet complexe (langage de programmation ou framework de développement).

Un exemple d'un tel outil est le système de persistance Java Open Source JPOX (www.jpox.org).

Ce système permet, entre autre, de stocker des objets java sous une forme persistante telle que décrite dans les spécifications JDO 1 et JDO 2 (<http://jcp.org/aboutJava/communityprocess/first/jsr012/>).

L'intérêt de ce produit est de fournir un langage de requêtage pour accéder aux objets stockés (pouvant donc représenter des tables spatiales), langage qui est indépendant des SGBD.

Une des fonctionnalités de ce langage de requêtage est la possibilité d'effectuer des requêtes spatiales sur les objets stockés et d'utiliser le résultat de ces requêtes.

JPOX support les 3 systèmes étudiés et se base sur la norme OGC pour le modèle de données spatiales.

On peut alors écrire des requêtes spatiales indépendantes des systèmes.

Exemples (suppose que la classe SampleLineString représentant une ligne a été définie comme objet stocké):

L'exemple suivant montre l'utilisation d'une fonction spatiale (length) dans le filtre d'une requête :

```
Double limit = new Double(100.0);  
Query query = pm.newQuery(SampleLineString.class, "geom != null &&  
Spatial.length(geom) < :limit");  
List list = (List) query.execute(limit);
```

L'exemple suivant montre une fonction spatiale pointN, pour extraire un point d'une ligne et en faire un objet Geometry :

```
query = pm.newQuery(SampleLineString.class, "id == :id");
query.setResult("Spatial.pointN(geom, 2)");
query.setUnique(true);
Geometry point = (Geometry) query.execute(new Long(1001));
```

L'exemple suivant montre comment utiliser des fonctions spatiales imbriquées dans une requête JPOX (Lister toutes les lignes dont le point final est égal au point créé par le programme :

```
Point point = new Point("SRID=4326;POINT(110 45)");
Query query = pm.newQuery(SampleLineString.class, "geom != null &&
Spatial.equals(Spatial.endPoint(geom), :point)");
List list = (List) query.execute(point);
```

Cet outil Java semble donc être une alternative intéressante pour garantir le portage des requêtes spatiales entre les 3 systèmes.

15.Restrictions

Chacun des systèmes étudiés apporte quelques restrictions concernant l'utilisation des données spatiales ou l'utilisation des fonctionnalités des systèmes.

Les chapitres sur les SRS ou les index listent quelques unes de ces restrictions en plus des chapitres suivants.

15.1.MySQL

La restriction majeure concernant MySQL est l'impossibilité d'utiliser le moteur transactionnel InnoDB avec les données spatiales.

MySQL met à disposition différents moteurs de stockage et de gestion des tables: MyISAM (le plus utilisé, historique), InnoDB (moteur transactionnel, ACID (atomicité, cohérence, isolation, durabilité), Memory (mémoire), Archive, Federated, etc.

Les transactions sont au cœur des SGBD et cette restriction concernant l'usage des données spatiales limite l'intérêt de MySQL.

Les autres restrictions concernent le manque de fonctionnalités spatiales de MySQL par rapport à celles définies par la norme OGC.

15.2.PostGIS

Une des restrictions de PostGIS, compte tenu de la richesse de ses fonctionnalités, est la non-gestion des données géocentriques, c'est à dire des coordonnées exprimées en longitudes latitudes. Il est alors très délicat de travailler avec des données s'étendant sur de grandes surfaces et d'obtenir des calculs de surfaces fiables.

Il faut passer par des fonctions permettant de simuler cette gestion, par exemple en découpant les polygones à cheval sur la ligne de date en deux polygones.

Certaines fonctions permettent de calculer des distances sur un sphéroïde, mais un véritable support pour des coordonnées exprimées entre 0 et 360° est manquant, ainsi que la gestion de

la ligne de dates (polygones s'étendant de part et d'autre de la longitude 180°).

15.3.Oracle

Oracle impose des restrictions concernant le sens de parcours des points formant des polygones. Il faut que ce sens soit le sens trigonométrique pour que les polygones soient valides et que les fonctions spatiales agissent correctement sur ces types.

La gestion des coordonnées géocentriques impose également des restrictions listées au chapitre 7.3.1.

La création d'index spatiaux est également obligatoire pour utiliser nombre de fonctions spatiales, mais cette création n'est pas automatique lorsqu'on crée une table avec une ou plusieurs colonnes spatiales. Il faut donc veiller à créer des index spatiaux de façon systématique pour chaque table stockant des données spatiales.

16.Accès externes aux systèmes

Le but de cette étude est de comparer les 3 systèmes de base de données en restant au plus près de ces systèmes pour éviter d'éventuels effets de bord dus à des applications tierces.

Cependant, un système est d'autant plus facilement utilisable qu'on peut y accéder par différents moyens, programmes, langages de programmation.

Les chapitres suivants listent quelques moyens possibles pour accéder aux systèmes.

Note: le site [Freegis.org](http://freegis.org) (<http://freegis.org>) liste la plupart des outils pouvant se connecter aux base de données PostGIS et MySQL.

16.1.MySQL

La cartouche spatiale de MySQL est la plus récente des 3 systèmes et la moins complète. Logiquement, il existe peu de solutions pour accéder à la partie spatiale de MySQL. En voici quelques unes (liste non exhaustive):

- OGR (<http://www.gdal.org/ogr>), bibliothèque Open Source de transformation de données vectorielles propose une connexion à MySQL et permet la lecture et l'écriture.
- FME, (<http://www.safe.com/>), logiciel de transformation de formats, supporte MySQL en lecture/écriture.
- shp2mysql, mysql2shp (<http://sourceforge.net/projects/shapetools>): programmes en ligne de commande ou en Perl pour convertir des données depuis/vers le format ESRI Shapefile.
- kvwmap (<http://kvwmap.geoinformatik.uni-rostock.de/index.php/Hauptseite>), serveur et client SIG web écrit en PHP, basé sur MapServer, supportant MySQL.
- MyWMS (<http://www.bt-gis.de/mediawiki/index.php/MyWMS>), un serveur WMS basé sur MySQL.

L'accès a MySQL lui-même, par contre, est possible depuis quasiment tous les grands langages de programmation. Il existe des API pour C/C++, Java, Python, Ruby, PHP, Delphi, Perl, etc. Il est possible d'inclure un serveur MySQL dans un programme sous la forme d'une bibliothèque.

16.2. PostGIS

PostGIS est actuellement la plus avancées des solutions libres pour stocker et manipuler des objets spatiaux dans une base de données (PostgreSQL).

La plupart des projets Open Source SIG peuvent afficher des données provenant de PostGIS. Certains peuvent mettre à jour des données dans PostGIS. La liste suivante montre quelques exemples de solutions (Open Source et propriétaires) pour se connecter à PostGIS :

- uDig (<http://udig.refractions.net>), par les auteurs de la cartouche PostGIS. SIG écrit en Java, pouvant lire et écrire dans PostGIS.
- Jump, OpenJump, gvSIG, Kosmo, DeeJump etc... Les SIG Java basés sur Jump (<http://www.vividsolutions.com/JUMP/>) peuvent pour la plupart lire/écrire dans PostGIS.
- Grass (<http://grass.itc.it/grass60/>). Actuellement le plus avancé et le plus riche des SIG libres, Grass se connecte directement à PostGIS pour traiter les données spatiales.
- MezoGIS : programme écrit en GTK+, il se veut un outil d'analyse spatiale pour PostGIS en restant très proche de SQL (édition graphique de requêtes spatiales). C'est un outil très pratique pour manipuler et tester les données spatiales dans PostGIS.
- Quantum GIS (<http://qgis.org>) est un SIG basé sur Qt permettant de lire/écrire des données PostGIS. Il permet également d'importer des shapefiles dans PostGIS.
- MapServer (<http://mapserver.gis.umn.edu>) Moteur cartographique le plus utilisé dans les applications de WebMapping, MapServeur peut se connecter directement à une base PostGIS pour afficher les données spatiales et leurs attributs.
- OGR (<http://www.gdal.org/ogr>), bibliothèque Open Source de transformation de données vectorielles propose une connexion à PostGIS et permet la lecture et l'écriture.
- GeoTools (<http://www.geotools.org/>), bibliothèque Java de manipulation de données spatiales, peut lire et écrire dans une base PostGIS
- Les serveurs WMS/WFS les plus populaires : Deegree, GeoServer ;
- ESRI ArcGIS (<http://www.esri.com>), le plus vendu des SIG peut se connecter à PostGIS par l'extension payante Data Interoperability (développée par Safe Software).
- FME, (<http://www.safe.com/>), logiciel de transformation de formats, supporte PostGIS en lecture/écriture
- etc, etc.

16.3. Oracle

Oracle est actuellement le leader mondial des bases de données (dans le monde des éditeurs payants).

Il existe une très grande quantité de programmes et de langages pouvant accéder à une base Oracle.

Concernant les données spatiales, la plupart de éditeurs de SIG offrent une connexion sur Oracle Spatial ou Locator. Quelques projets Open Source permettent également de se connecter à une base Oracle :

- uDig (<http://udig.refractions.net>), par les auteurs de la cartouche PostGIS. SIG écrit en

Java, pouvant lire depuis Oracle.

- MapServer (<http://mapserver.gis.umn.edu>) Moteur cartographique le plus utilisé dans les applications de WebMapping, MapServer peut se connecter directement à une base Oracle pour afficher les données spatiales et leurs attributs.
- OGR (<http://www.gdal.org/ogr>), bibliothèque Open Source de transformation de données vectorielles propose une connexion à Oracle et permet la lecture et l'écriture.
- GeoTools (<http://www.geotools.org/>), bibliothèque Java de manipulation de données spatiales, peut lire et écrire dans le format Oracle
- ESRI ArcGIS (<http://www.esri.com>), le plus vendu des SIG peut se connecter à Oracle Spatial et Locator
- FME, (<http://www.safe.com/>), logiciel de transformation de formats, supporte Oracle en lecture/écriture.
- MapInfo (<http://www.mapinfo.com/>), logiciel SIG pour Windows
- etc...

17. Outils d'administration

L'administration d'un système de base de données est une tâche importante dans un projet mettant en œuvre un SGBD. Il peut s'agir de tâches de sauvegarde, de réplication (copie d'une base vers une ou plusieurs autres, en vue d'une sauvegarde ou d'une amélioration de performance), de répartition de charge (répartir une même base de données sur plusieurs machines pour profiter de meilleures performances), etc.

Un système complet doit mettre à disposition ces outils d'administration et les rendre le plus facile possible.

17.1. MySQL

Il existe de nombreux outils d'administration fournis avec MySQL lors de l'installation. Il existe également des outils externes, développés par des sociétés ou par des projets OpenSource.

MySQL met notamment à disposition un outil graphique d'administration de bases de données MySQL: MySQL Administrator. Il permet d'effectuer toutes les tâches d'administration depuis une console graphique (sauvegarde, audit, réplication, optimisation, etc.)

MySQL propose également des utilitaires en ligne de commande permettant de réaliser des tâches d'administration.

17.1.1. mysqldump

MySQL propose un utilitaire de sauvegarde: mysqldump. Il permet de réaliser la sauvegarde d'une ou plusieurs bases MySQL. Les formats de sortie sont SQL, Text, CSV, XML.

Les paramètres du programme permettent de contrôler finement le type de sauvegarde à effectuer.

Il existe d'autres utilitaires de sauvegarde, comme par exemple mysqlhotcopy pour sauvegarder et restaurer de façon efficace des tables gérées par le moteur MyISAM.

17.1.2.mysqlimport

Les données générées par le programme de sauvegarde ou provenant de sources externes peuvent être importées dans MySQL avec l'outil mysqlimport.

17.2.PostGIS

Parmi les 3 systèmes testés, PostgreSQL/PostGIS est celui qui offre le moins d'outils d'administration.

Le système met à disposition des utilitaires en ligne de commande pour sauvegarder et restaurer les bases de données. Le format de sauvegarde, ainsi que les éléments sauvegardés sont contrôlés par les paramètres du programme.

17.2.1.pg_dump, pg_restore

Les programmes pg_dump et pg_restore permettent respectivement de sauvegarder et restaurer une base de données PostgreSQL ainsi que la partie spatiale PostGIS.

Les paramètres de ces programmes permettent de contrôler le format et le type des objets sauvegardés.

Il faut noter que les sauvegardes peuvent être réalisées à chaud, c'est à dire sans devoir arrêter le système.

17.2.2.Slony-I

PostgreSQL propose un outil de réplication de bases de données appelé Slony-I.

Cet outil permet de répliquer une base maître vers plusieurs bases esclaves, soit pour des raisons de sauvegarde, soit pour des raisons de performance.

Cet outil n'est pas facile à mettre en place et requiert des compétences d'administrateur de bases de données.

17.3.Oracle

Oracle 10.2g propose une série de programmes et d'utilitaires pour les tâches d'administration.

Le programme Oracle Enterprise Manager, notamment, permet d'effectuer toutes les tâches d'administration, de sécurité, d'audit, etc. sur des instances Oracle à partir d'un navigateur web:

- Audit des bases de données
- Fichiers de logs
- Sauvegardes et restauration
- Réplication
- Mesure et réglage de performance
- etc.

Oracle recommande l'utilisation de cet outil pour les tâches d'administration.

Il existe de nombreux guide et une documentation complète sur toutes les tâches d'administration touchant a une base Oracle, par exemple Oracle Database Administrator's Guide (http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14231/toc.htm)

18. Conclusion

Cette étude met en avant les caractéristiques suivantes pour chacun des systèmes:

- **MySQL Spatial** est encore un projet jeune, en cours de développement et n'implémente pas de façon complète la norme. Il permet néanmoins le stockage et l'indexation d'objets spatiaux et offre de très bonnes performance pour les requêtes mettant en œuvre des objets spatiaux. A terme, si MySQL implémente la norme OGC, il représentera un concurrent sérieux a PostGIS. Son avantage est cependant une communauté de développeurs et de support très active, une très bonne intégration dans les projets Web (base de données la plus utilisée dans le monde pour ce type de projets).
- **Oracle Locator** ne fournit qu'un sous-ensemble des fonctions définies par la norme OGC. Le système permet le stockage et l'indexation de données spatiales, ainsi que le requetage d'objets basé sur des opérateurs spatiaux. Il ne permet pas de créer de nouveaux objets et ne contient aucun module avancé comme ceux proposés par Spatial.

- **Oracle Spatial** offre une très grande quantités de fonctions spatiales. Il implémente la norme OGC complètement mais ne respecte pas le nommage des fonctions dans la norme. Ceci nuit considérablement a la portabilité des requêtes spatiales.

Son support des images, de la topologie, du géocodage, du calcul d'itinéraire en fait un logiciel SIG complet, pouvant s'interfacer avec un grand nombre de SIG du commerce. Ses coûts d'utilisation et de support sont élevés et sa maintenance compliquée.

- **PostGIS** implémente la norme OGC et offre un grand nombre de fonctionnalités spatiales. Il n'offre pas autant de fonction qu'Oracle. La dernière version de PostGIS, 1.2.1, propose de nouvelles fonctions comme le support partiel de la norme SQL-MM, l'optimisation de certaines requêtes spatiales, etc. La communauté Open Source de ce projet est très active, tant pour le développement que pour l'animation des forums. Le temps de résolution d'éventuels bugs est très court et peu de problèmes restent sans réponse. Des développements sont en cours concernant le support de la topologie dans PostGIS et des réflexions sont menées pour l'intégration et le support des rasters.

Des modules externes a PostGIS permettent de gérer le calcul d'itinéraires (pgRouting) et le géocodage.

Dans le cadre de la rationalisation des moyens informatiques au sein du CNES et en prenant en compte les besoins exprimés sur le projet Pleiades, **PostGIS paraît un bon compromis entre richesse des fonctionnalités, facilité de mise en oeuvre, dynamisme de la communauté, et performance.**

MySQL Spatial est encore jeune et manque de fonctionnalité pour pouvoir couvrir les besoins actuels et futurs des projets mettant en œuvre des données spatiales. Le système dispose cependant d'atouts lors de la mise en place de projets Web.

Oracle Spatial est un SIG complet, mais lourd à mettre en œuvre et cher à l'utilisation, à la différence des autres systèmes qui n'ont pas de coût de licences et d'utilisation.

Les différences entre les 3 systèmes et leur respect de la norme OGC sont tel que les requêtes spatiales sont propres à chaque système et difficilement portables entre les systèmes, à l'exception de quelques requêtes simples qui s'écrivent de la même façon dans MySQL et dans PostGIS.

19. Annexe 1: Information sur les systèmes

19.1. MySQL

- Site Web: <http://mysql.com>
- Documentation: <http://dev.mysql.com/doc/refman/5.1/en/index.html>

19.2. PostGIS

- Site Web: <http://postgis.refrations.net>
- Documentation: <http://postgis.refrations.net/docs/>

19.3. Oracle Spatial

- Site Web: <http://www.oracle.com/technology/products/spatial/index.html>
- Documentation: http://download-uk.oracle.com/docs/html/B14255_01/toc.htm

19.4. Autres ressources

Documentation JPOX: http://www.jpox.org/docs/1_2/jdo/jdoql_spatial_methods.html