



## Conseil Régional d'Aquitaine - Délégation TIC

### - Phase 1 - Rapport d'analyse du MCD et création de la base

**Gr@ce**

**27 mai 2010**

Réf. Consultation : 2010IA000S03060000  
Réf. Makina Corpus : MC-2010-0293

**Sylvain Beorchia**

+33 6 10 65 74 00

**Makina Corpus**

8, place André Abbal

31100 Toulouse

[www.makina-corpus.com](http://www.makina-corpus.com)



---

## Table des Matières

<b>1</b>	<b><u>Validation du MCD</u></b> .....	<b>4</b>
	[1.1] <u>Analyse du MCD existant</u> .....	4
	[1.2] <u>Modifications et remarques sur le MCD</u> .....	4
	[1.3] <u>MCD final</u> .....	6
<b>2</b>	<b><u>Designer</u></b> .....	<b>7</b>
<b>3</b>	<b><u>Création de la base de données</u></b> .....	<b>7</b>
	[3.1] <u>Pré-requis</u> .....	8
	[3.2] <u>Création de la base de données</u> .....	8
	[3.3] <u>Création de la structure de la base</u> .....	9
	[3.4] <u>Remplissage de la base</u> .....	10
	[3.5] <u>Exemples d'exploitation de la base</u> .....	10
<b>4</b>	<b><u>Annexe 1 - description du MCD</u></b> .....	<b>11</b>

# 1 Validation du MCD

## [1.1] Analyse du MCD existant

---

Le MCD établi par la délégation TIC de la région Aquitaine a été fait à partir de plusieurs sources de données et informations récoltées. De premier abord il semble globalement satisfaisant et pérenne.

De par les informations récupérées lors des réunions, et en recoupant avec des MCD réalisés pour les réseaux télécom par des ressources internes (makina corpus), on retrouve une similitude dans les tables employées et les relations établies. L'absence de jeu de données complet limite la vérification des cardinalités et des relations. Au fur et à mesure de l'intégration des données, le modèle de données pourra être vérifié à nouveau et surtout testé.

Le MCD est donc validé dans sa globalité jusqu'aux tests finaux. Les paragraphes suivants décrivent les modifications qui sont apportées.

## [1.2] Modifications et remarques sur le MCD

---

### [1.2.1] Identifiants

Pour garantir la pérennité des données ainsi que leur intégrité (notamment lors des phases de mise à jour des données), un identifiant unique doit être attribué à chaque objet. Cet identifiant constitue la clef primaire des objets et sera généré lors du premier import des données des partenaires. Les données seront ensuite redistribuées aux partenaires de manière à ne pas perdre les identifiants.

Les mises à jour se feront alors sur des objets dont l'identifiant est connu sur la base centrale, et la synchronisation pourra se faire sans problème.

Il a été décidé d'établir les identifiants selon le système suivant :

`code_insee_diminutif_table_numéro_aléatoire`

Le diminutif table étant son nom raccourci ou ses initiales pour les noms composés.

Les collectivités fournisseuses de données étant localisées géographiquement, il sera possible par ce système savoir d'où proviennent les objets (indication non précise, juste indicative).

Les ID des tables sont donc des champs de type varchar.

## **[1.2.2] Table légende**

Plusieurs tables utilisent des listes de valeurs pour indiquer une caractéristique. Ces listes étaient établies de la manière suivante (exemple) :

- 1 - Bon état
- 2 - Etat Moyen
- 3 - Etat médiocre

Plutôt que de stocker des chaînes de caractères contenant et le chiffre et la description, une table Légende est ajoutée au MCD. Cette dernière contient 4 colonnes. Un exemple de contenu est le suivant :

<b>id</b>	<b>groupe</b>	<b>num_choix</b>	<b>libelle_choix</b>
1	etat	1	très abîmé
2	etat	2	abîmé
3	etat	3	correct
4	etat	4	bien
5	etat	5	neuf
6	etat	0	inexistant (en projet)

Ainsi, dans une application connectée à la base de données, l'affichage des valeurs pourra se faire selon le SQL suivant (on veut par exemple afficher l'état d'un fourreau) :

```
SELECT libelle_choix FROM legende WHERE num_choix = xx AND groupe = 'etat';
```

Les listes ne sont pas nominatives par rapport à une table. En effet, une même liste peut-être utilisée pour plusieurs tables. Ceci permet d'optimiser et d'homogénéiser les libellés, et ainsi éviter les erreurs de saisies. Les personnes qui mettront à jour la base rempliront eux des numéros en toute connaissance des correspondances. Si un numéro n'a pas de correspondance dans la table légende, alors un avertissement sera reporté au moment de l'import ou de la mise à jour des données.

Le script SQL de remplissage de cette table est inclus dans le fichier « grace\_sig.sql ». Il est établi à partir des descriptions fournies par la délégation TIC de la région Aquitaine. Bien sûr cette table pourra évoluer dans le temps.

### **[1.2.3] Formats et homogénéité**

- Les dates devront être saisies dans le format suivant (dans les données entrantes) :

jj/mm/aaaa

- Les champs notés comme hyperliens, sont des champs de type varchar. Libre à l'application qui exploitera les données de les afficher en tant que liens.
- Tous les champs géométriques sont nommés « geom », quelque soit leur table d'appartenance. Ceci rend plus facile leur exploitation future (ex: définition des couches dans un mapfile, on sait que tous les champs sont nommés geom).
- Les noms des colonnes sont homogénéisés de manière à respecter une certaine logique de création des tables :
  - lorsque le nom de la table est composé d'un seul mot, les noms des colonnes sont suffixés du nom de la table, ou si le nom est long, d'une contraction explicite du nom (ex: chambre => chbre).
  - lorsque le nom de la table est composé de plusieurs mots, les noms des colonnes sont suffixés des initiales du nom de la table.
  - les champs de jointures ne respectent pas cette règle. Ils portent le nom de la clef de la table jointure.
  - quelques exceptions sont faites pour certaines tables possédant plusieurs jointures vers une autre table (artère/nœud), ou des jointures sur elle même (local technique).
- Les relations dont les tables ne sont pas présentes ont été supprimées (offre, tube). Elles pourront être rajoutées au besoin.

## **[1.3] MCD final**

Un certain nombre de modifications ont été apportées au MCD original. Le MCD final intégrant toutes ces modifications est disponible dans le projet SQLPA. Il est inutile de décrire toutes ces modifications dans le présent document car il s'agit parfois d'erreur d'étourderie, de mauvais types de champs ou encore de relations manquantes sur le MCD. Tout ceci est corrigé sur le MCD final qui devient le MCD de référence. Le fichier projet SQLPA est disponible pour consulter ce MCD en détail.

## 2 Designer

Le designer de base de données SQL Power Architect (SQLPA) a été choisi. Il est disponible dans une version OpenSource, bien que des versions « Entreprise » existent moyennant finances.

Ce designer a la particularité d'être simple et ergonomique. Les fonctionnalités sont peu nombreuses mais efficaces et correspondantes aux besoins du projet Gr@ce. Outre le fait que l'utilisateur puisse y décrire précisément son modèle de données et le générer ensuite dans le système de base de données de son choix, il est possible de faire des analyses simples sur les données contenues dans les tables (graphiques, pourcentages...).

Son point négatif est qu'il ne gère pas les champs géométries nécessaires pour le projet. Nous contournons cette limitation en produisant un script SQL complémentaire qui devra être lancé à la main après l'exécution du script principal produit par SQLPA. Cependant, il est fort probable que dans les mois à venir ce manquement soit comblé par une mise à jour du logiciel (en provenance de la communauté, ou d'un privé produisant l'effort).

Le MCD de la base Gr@ce a été recréé sous SQL Power Architect, et le fichier projet a été livré à la délégation TIC de la région Aquitaine.

Une formation rapide sur l'outil à été délivrée afin d'assurer sa bonne prise en main. L'installation du logiciel est facile sous Windows et il est ensuite accessible dans le menu démarrer. Sous Linux, il est nécessaire de télécharger le code source, et de la lancer avec la commande :

```
java -jar architect-0.9.16/architect.jar
```

## 3 Création de la base de données

Les opérations décrites ci-dessous prennent pour base un système linux (Debian). Sous Windows, il convient d'utiliser l'outil pgAdmin pour exécuter ces opérations.

De plus, si la personne en charge de la création de la base a des habitudes autres, il est libre de les exécuter à sa manière. Dans ce cas, seuls les scripts « grace.sql » et « grace\_sig.sql » devront au moins être lancés sur la base.

## [3.1] Pré-requis

---

PostgresSql et sa cartouche spatiale PostGis doivent être installés sur le serveur. Sous linux (Debian), l'installation peut se faire via les paquets :

```
sudo aptitude install postgresql-8.4
sudo aptitude install postgresql-8.4-postgis
```

La version antérieure (8.3) peut également être utilisée (dans le cas où le serveur possède déjà une base Postgres).

Par défaut, l'installation de la base Postgres crée un utilisateur système Postgres. Ce dernier permet d'effectuer des opérations d'administration sur la base (droits administrateurs). Cependant, dans une application se servant de la base de données, on va préférer créer un nouvel utilisateur avec des droits que l'on pourra paramétrer plus spécifiquement en fonction des besoins :

```
# On bascule vers l'utilisateur Postgres
sudo su - postgres

# on lance l'environnement SQL
psql

> CREATE USER gisuser WITH PASSWORD 'password_a_definir';
```

On va donner par la suite à cet utilisateur les droits d'accès aux tables.

## [3.2] Création de la base de données

---

La procédure décrite ci-dessous concerne un serveur linux (Debian). Pour Windows, il convient au DBA d'utiliser pgAdmin (ou un autre outil).

```
# On bascule vers l'utilisateur Postgres
sudo su - postgres

# création de la base
createdb grace

# ajout du langage plpgsql à la base
createlang plpgsql grace

# ajout des composants spatiaux (v8.3)
psql -d grace -f /usr/share/postgresql-8.3-postgis/lwpostgis.sql
psql -d grace -f /usr/share/postgresql-8.3-postgis/spatial_ref_sys.sql
```

```
# ou (v.8.4)
psql -d grace -f /usr/share/postgresql/8.4/contrib/postgis.sql
psql -d grace -f /usr/share/postgresql/8.4/contrib/spatial_ref_sys.sql
```

La base est maintenant prête à accueillir ses tables et données.

### [3.3] Création de la structure de la base

Le script « grace.sql » généré par SQLPA peut être lancé. Sous linux, la commande est la suivante :

```
# basculer en utilisateur postgres
sudo su - postgres
# exécution du script
psql -d grace -f /chemin_vers_script/grace.sql
```

*Remarque : si la structure de la base a déjà été créée, et que l'on souhaite recommencer, il faudra alors exécuter le script « grace\_pre.sql » qui assure la destruction du modèle.*

Si tout se déroule correctement, aucune erreur ne doit être affichée, et un COMMIT final doit avoir été exécuté automatiquement.

Le script « grace\_sig.sql » doit également être lancé afin d'ajouter les colonnes géométries aux tables concernées (fourreau, cable, sous\_tubage, artere, noeud) :

```
# exécution du script
psql -d grace -f /chemin_vers_script/grace_sig.sql
```

Si tout se déroule correctement, aucune erreur ne doit être affichée.

L'exécution du script s'est faite sous l'utilisateur Postgres. Il est donc nécessaire de donner les droits à l'utilisateur Gisuser:

```
psql -d grace -c "ALTER DATABASE grace OWNER TO gisuser"
psql -d grace -c "`psql -F " " -Atd grace -c "select 'ALTER TABLE ' ||tablename||'
OWNER TO gisuser;' from pg_tables where (tablename not like 'pg_%') and (tablename
not like 'sql_%') order by tablename`"
```

Afin de vérifier le bon déroulement de ces scripts, une consultation de la base peut être effectuée en utilisant l'outil PgAdmin par exemple.

## [3.4] Remplissage de la base

---

Ce point fait l'objet des phases 2 et 3 du projet, il sera donc détaillé en temps voulu.

## [3.5] Exemples d'exploitation de la base

---

Le MCD a été conçu de manière à pouvoir consulter de manière simple les objets de la base. Ci-dessous sont listées des phrases SQL permettant différentes opérations :

- récupérer les fourreaux rattachés à une artère :  
`SELECT * FROM fourreau WHERE id_artere = 'mon_id';`
- récupérer les fourreaux de la rue « mon\_id\_rue » :  
L'information de la rue est contenue dans la table artere, il faut donc imbriquer deux requêtes :  
`SELECT * FROM fourreau WHERE id_artere in (SELECT id_artere FROM artere WHERE id_com_insee_artere = 'mon_id_rue') ;`
- compter le nombre de fourreau de l'artère « mon\_id\_artere » :  
`SELECT count(*) FROM fourreau WHERE id_artere = 'mon_id_artere';`
- récupérer le local technique père du local technique « mon\_id\_lt » :  
`SELECT id_lt_pere FROM local_technique WHERE id_lt = 'mon_id_lt';`
- récupérer les fils du local technique « mon\_id\_lt » :  
`SELECT id_lt FROM local_technique WHERE id_lt_pere = 'mon_id_lt' ;`

Ce ne sont que de simples exemples mais qui peuvent permettre d'accéder à n'importe quelle informations dans la base.

## 4 Annexe 1 - description du MCD

Le MCD sous forme tabulaire est fourni en annexe (grace.pdf).

