



**MINISTÈRES
TRANSITION ÉCOLOGIQUE
COHÉSION DES TERRITOIRES
MER**

*Liberté
Égalité
Fraternité*

**Secrétariat général
Service du numérique**

DOCUMENTATION UTILISATEUR



ASGARD

Système de gestion des droits pour
PostgreSQL

SOUS-DIRECTION MÉTHODES ET SERVICES DE PLATEFORME
DÉPARTEMENT SOCLE
GROUPE SOCLE GÉOMATIQUE
MÉL : GSG.DS.MSP.SNUM.SG@DEVELOPPEMENT-DURABLE.GOUV.FR

SOUS-DIRECTION USAGES NUMÉRIQUES ET INNOVATION
DÉPARTEMENT RELATION CLIENT
MÉL : DRC.UNI.SNUM.SG@DEVELOPPEMENT-DURABLE.GOUV.FR

15 SEPTEMBRE 2020

DOCUMENT

OBJET

Documentation utilisateur d'ASGARD, système simplifié et automatisé de gestion des droits pour PostgreSQL.

ASGARD a été élaboré conjointement par le groupe socle géomatique et le département relation client du SNUM dans le cadre des travaux du groupe de travail interministériel PostGIS (MAA, MTE/MCTRCT/Mer).

L'extension PostgreSQL ASGARD (© République Française, 2020) est publiée sur le site geoinformations.developpement-durable.gouv.fr sous licence CeCILL-B.

HISTORIQUE DES VERSIONS

V.0	2020.05.27	AF Rédaction initiale. Réf. : ASGARD 0.6.0.
V.1	2020.06.03	LL Relecture et compléments. Réf. : ASGARD 0.6.2.
V.2	2020.07.08	LL Adaptation aux évolutions d'ASGARD, modification de la doctrine sur la délégation de la gestion des rôles. Réf. : ASGARD 0.7.1.
V.3	2020.09.02	LL Adaptation aux évolutions d'ASGARD. Réf. : ASGARD 1.0.0.
V.4	2020.09.15	LL Adaptation aux évolutions d'ASGARD. Réf. : ASGARD 1.1.0.

AUTEURS

Rédaction de la documentation et développement de l'extension PostgreSQL ASGARD : Alain Ferraton (groupe socle géomatique) & Leslie Lemaire (département relation client).

RELECTURE

Michel Zevort, groupe socle géomatique. Membres du groupe de travail PostGIS, avec notamment les contributions de Philippe Allamand, Michel Auzanneau, Erick Bouaillon, Dominique Choquet, Jean-François Pion et Luc Sommermeyer.

TABLE DES MATIÈRES

DOCUMENT	2	III — UTILISATION PRATIQUE D'ASGARD	20
OBJET	2	III.1 — VUE GESTION_SCHEMA_USR	20
HISTORIQUE DES VERSIONS	2	III.1.A. ACCÈS À LA VUE	22
AUTEURS	2	III.1.B. DÉLÉGUER LA GESTION DES DROITS	23
RELECTURE	2	III.1.C. CAS PARTICULIER D'UN SCHÉMA DONT LE PRODUCTEUR EST UN SUPER-UTILISATEUR	26
I — INTRODUCTION	6	III.1.D. PGADMIN ET L'ÉDITION DES VUES	26
I.1 — CONTEXTE	6	III.2 — CHAMPS DE LA TABLE DE GESTION	27
I.2 — PRÉSENTATION SUCCINCTE	6	III.2.A. BLOC	27
I.3 — GÉNÉRALITÉS SUR LES DROITS ET RÔLES	7	III.2.B. NOMENCLATURE	32
I.3.A. PRÉROGATIVES DU PROPRIÉTAIRE	7	III.2.C. NIV1	33
I.3.B. RÔLE DE CONNEXION ET RÔLE DE GROUPE	8	III.2.D. NIV1_ABR	33
I.3.C. ATTRIBUTS CREATEROLE ET CREATEDB	8	III.2.E. NIV2	33
II — GESTION DES DROITS AVEC ASGARD	10	III.2.F. NIV2_ABR	33
II.1 — PRINCIPES	10	III.2.G. NOM_SCHEMA	34
II.2 — NOMMAGE	12	III.2.H. CREATION	34
II.2.A. RÈGLE (OPTIONNELLE) DE NOMMAGE DES RÔLES DE CONNEXION	12	III.2.I. PRODUCTEUR	34
II.2.B. RÈGLE (OPTIONNELLE) DE NOMMAGE DES RÔLES DE GROUPE	13	III.2.J. EDITEUR	35
II.2.C. CARACTÈRES INTERDITS	13	III.2.K. LECTEUR	36
II.3 — RÔLES D'ASGARD	13	III.3 — CRÉATION DE SCHÉMAS DE LA NOMENCLATURE NATIONALE	37
II.3.A. G_CONSULT — CONSULTATION DES DONNÉES PUBLIQUES	14	III.3.A. CRÉATION D'UN SOUS-ENSEMBLE DE SCHÉMAS DE LA NOMENCLATURE	39
II.3.B. G_ADMIN — ADMINISTRATION DE LA BASE	16	III.4 — CRÉATION DE NOUVEAUX SCHÉMAS	39
II.3.C. CONSULT.DEFAUT — RÔLE DE CONNEXION GÉNÉRIQUE	18	III.5 — SUPPRESSION DE SCHÉMAS	41
II.3.D. G_ADMIN_EXT — RÔLE TECHNIQUE	19	III.6 — MODIFICATIONS DE SCHÉMAS	43
II.3.E. CAS DE RÔLES PRÉ-EXISTANTS	19	III.6.A. RENOMMER UN SCHÉMA	43

III.6.B. MODIFIER LE PROPRIÉTAIRE D'UN SCHÉMA	43	V.2.A. CONSIDÉRATIONS SUR LA DÉFINITION DE LA CONNEXION	85
III.6.C. MODIFIER LES PRIVILÈGES	44	V.2.B. CONNEXION DÉFINIE PAR UN « SERVICE »	91
III.7 — CRÉATION D'UN OBJET DANS UN SCHÉMA	46	V.2.C. UTILISATION DES FICHIERS INI	93
III.8 — MODIFICATION DES OBJETS	46	V.3 — UTILISATION COURANTE	94
III.9 — OBJETS NON PRIS EN CHARGE	48	VI — INSTALLATION D'ASGARD	96
III.10 — MODIFICATION DES RÔLES	48	VI.1 — COMPATIBILITÉ	96
III.11 — SAUVEGARDE ET RESTAURATION DE LA BASE	50	VI.2 — LIVRABLES	96
III.12 — NETTOYAGE DES DROITS ET RÉOLUTION DES PROBLÈMES	50	VI.3 — INSTALLATION DE L'EXTENSION	96
IV — FONCTIONS UTILITAIRES	52	VI.4 — INITIALISATION DE LA TABLE DE GESTION	98
IV.1 — PRÉSENTATION	52	VI.5 — MISE À JOUR DE L'EXTENSION	101
IV.2 — ASGARD_DEPLACE_OBJ	54	VI.6 — DÉINSTALLATION DE L'EXTENSION	101
IV.3 — ASGARD_INITIALISE_SCHEMA	57	VI.7 — DÉSACTIVER ASGARD ?	102
IV.4 — ASGARD_INITIALISE_OBJ	61	VII — MISE EN ŒUVRE SUR UN EXEMPLE	104
IV.5 — ASGARD_NETTOYAGE_ROLES	63	VII.1 — DESCRIPTION DU CAS PRATIQUE	104
IV.6 — ASGARD_SORTIE_GESTION_SCHEMA	64	VII.2 — PRÉPARATION DE LA BASE DE DONNÉES ET INSTALLATION D'ASGARD	105
IV.7 — ASGARD_INITIALISATION_GESTION_SCHEMA	65	VII.3 — CRÉATION DES RÔLES ET INITIALISATION D'ASGARD	107
IV.8 — ASGARD_ALL_LOGIN_GRANT_ROLE	67	VII.4 — MISE EN PLACE DES CONNEXIONS POSTGRESQL SOUS QGIS	118
IV.9 — ASGARD_IMPORT_NOMENCLATURE	69	VII.5 — FINALISATION DE LA CRÉATION DES SCHÉMAS ET IMPORT D'OBJETS	127
IV.10 — ASGARD_REAFFECTE_ROLE	72	VII.6 — MENU POUR QGIS	133
IV.11 — ASGARD_INITIALISE_ALL_SCHEMAS	77	ANNEXE A — FONCTIONS SUPPLÉMENTAIRES	135
IV.12 — ASGARD_DIAGNOSTIC	79	ASGARD_ADMIN_PROPRIETAIRE	135
V — MENU POUR QGIS	83	ASGARD_GRANT_TO_REVOKE	136
V.1 — PRINCIPE	83	ASGARD_SYNTHESE_PUBLIC	137
V.2 — PARAMÉTRAGE D'ASGARDMENU	85	ASGARD_SYNTHESE_PUBLIC_OBJ	138

ASGARD_SYNTHESE_ROLE	138	ASGARD_ROLE_TRANS_ACL	140
ASGARD_SYNTHESE_ROLE_OBJ	139	ANNEXE B — LISTE DES RÈGLES, PRINCIPES ET BONNES PRATIQUES	141

I — INTRODUCTION

I.1 — CONTEXTE

ASGARD¹ s'inscrit dans le cadre de travaux visant à généraliser l'usage des bases de données PostgreSQL comme solution de stockage du patrimoine de données géomatiques des services.

Il répond à deux objectifs :

- ◆ mettre à disposition des services qui démarrent sur PostgreSQL une solution de gestion des droits intégrée clé en main qui, sans nécessiter de compétences avancées sur PostgreSQL, est adaptée au contexte d'un serveur de données partagé, notamment du point de vue de la sécurité ;
- ◆ fournir à l'ensemble des services qui le souhaitent une solution de gestion des droits qui automatise les tâches d'administration répétitives sans pour autant restreindre l'usage des fonctionnalités natives de PostgreSQL.

I.2 — PRÉSENTATION SUCCINCTE

ASGARD prend la forme d'une extension pour PostgreSQL, qui met en œuvre :

- ◆ un modèle de gestion des droits simplifié, qui alloue par défaut des droits homogènes pour tous les objets d'un même schéma ;
- ◆ une classification des schémas en blocs fonctionnels avec des règles de nommage ;

1 « Automatic and Simplified GrAnting for Rights in Databases » (Octroi automatisé et simplifié pour les droits dans les bases de données). Référence aux [archées d'Asgard](#) et clin d'œil au projet OSHIMAE. Le dessin de couverture représente l'arbre monde (Yggdrasil) sur lequel reposent les neuf royaumes dont Asgard, royaume des Ases.

- ♦ une nomenclature nationale des schémas pour le bloc fonctionnel « consultation » – et éventuellement « référentiel » dans le futur – reprenant les deux niveaux de l'arborescence COVADIS pour la Géobase ;
- ♦ un couplage avec le plugin AsgardMenu pour QGIS, pour offrir aux utilisateurs un menu hiérarchique leur présentant le patrimoine PostgreSQL auquel ils ont accès.

La nomenclature des schémas et les règles de nommage sont une production antérieure du groupe de travail PostGIS, décrite dans le document *Préconisation de structuration des données en base PostGIS – niveau schéma*². ASGARD permet de les utiliser, mais ne l'impose pas. Une fonction de reprise permet par ailleurs d'intégrer des schémas existants dans le mécanisme de gestion des droits d'ASGARD.

I.3 — GÉNÉRALITÉS SUR LES DROITS ET RÔLES

Nous rappelons ici quelques notions fondamentales de la gestion des droits sous PostgreSQL.

I.3.A. PRÉROGATIVES DU PROPRIÉTAIRE

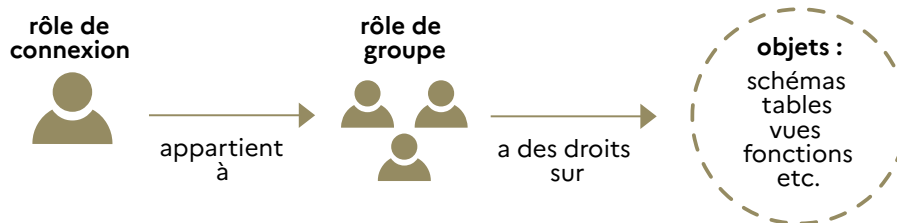
Le rôle désigné comme propriétaire d'un objet (schéma, table...) a nativement tous les droits sur cet objet, en particulier celui de le supprimer ou modifier sa définition (commande ALTER, ou CREATE OR REPLACE sur un objet pré-existant). Ces deux derniers droits sont intrinsèques à la propriété – ils ne peuvent être ni conférés à d'autres, ni révoqués.

PostgreSQL permet cependant au propriétaire d'attribuer d'autres privilèges aux rôles qu'il souhaite sur les objets qui lui appartiennent : droit d'accéder à l'objet (dans le cas d'un schéma, par exemple), de l'utiliser (par exemple pour une fonction), de voir ou de modifier les données qu'il contient (dans le cas d'une table ou assimilé)...

2 Document diffusé sur Géoinformations à l'adresse suivante : <http://www.geoinformations.developpement-durable.gouv.fr/-a3733.html>.

I.3.B. RÔLE DE CONNEXION ET RÔLE DE GROUPE

ASGARD adopte l'organisation habituelle suivante :



L'administrateur définit des rôles de connexion (attribut LOGIN), qui permettent aux utilisateurs de se connecter à la base, et des rôles de groupe (attribut NOLOGIN).

BONNE PRATIQUE. Les droits sur les objets sont exclusivement accordés aux rôles de groupe.

Un rôle de connexion ne reçoit pas de droit direct, mais il hérite des droits du ou des rôles de groupe auxquels il appartient.

I.3.C. ATTRIBUTS CREATEROLE ET CREATEDB

Comme l'attribut LOGIN mentionné au paragraphe précédent, les attributs CREATEROLE et CREATEDB peuvent être conférés à un rôle au moment de sa création ou, a posteriori, en modifiant sa définition³.

Un rôle disposant de l'attribut CREATEROLE peut gérer les rôles : octroyer ou révoquer l'appartenance d'un rôle à un autre, créer de nouveaux rôles, modifier les attributs des rôles existants et supprimer des rôles (tout ceci hors rôles super-utilisateurs).

Pour sa part, l'attribut CREATEDB est nécessaire pour créer des bases de données, modifier leur nom et leur propriétaire, ou encore les supprimer.

L'administrateur de données localisées (ADL) du service est pleinement légitime à disposer de ces deux attributs, cependant ils ne doivent pas être délégués à la légère à d'autres utilisateurs. En particulier, on retiendra qu'un rôle doté de CREATEROLE peut théoriquement agir sur tous les objets dont le propriétaire n'est pas un super-utilisateur simplement en se rendant membre de leur rôle propriétaire.

³ Par une commande ALTER ROLE.

Au contraire des droits sur les objets, un rôle n'hérite pas directement des attributs des rôles de groupe dont il est membre.

EXEMPLE. Si le rôle de connexion C est membre du rôle de groupe G, il doit explicitement endosser le rôle G pour bénéficier des attributs de ce dernier :

```
SET ROLE role_g ;
```

Il n'a alors plus accès à ses propres attributs jusqu'à ce qu'il réalise l'opération inverse :

```
SET ROLE role_c ;
```

... ou encore

```
RESET ROLE ;
```

Ainsi, il est souvent préférable de conférer directement les attributs CREATEDB et CREATEROLE aux rôles de connexion des utilisateurs concernés, d'autant que ceux-ci se doivent de rester très peu nombreux.

BONNE PRATIQUE. Pour autoriser à un utilisateur à créer des rôles ou des bases, on donne à son rôle de connexion l'attribut CREATEROLE ou, respectivement, l'attribut CREATEDB.

EXEMPLE. Pour habilitier le rôle de connexion jon.snow à gérer les rôles :

```
ALTER ROLE "jon.snow" WITH CREATEROLE ;
```

II — GESTION DES DROITS AVEC ASGARD

II.1 — PRINCIPES

Le mécanisme par défaut de PostgreSQL n'est généralement pas conforme à un comportement simplifié souhaité par les services. En effet, il implique des actions pour les créateurs d'objets s'ils souhaitent que d'autres utilisateurs aient accès à leurs objets. Lorsque cette attribution explicite de droits est réalisée à la main, elle peut rapidement devenir fastidieuse et source d'erreurs.

Avec ASGARD des droits standards sont appliqués automatiquement au moment de la création des objets dans les schémas et le propriétaire est ré-affecté de manière à rester cohérent avec le propriétaire du schéma.

RÈGLE IMPOSÉE PAR ASGARD⁴. Le propriétaire d'un schéma est le propriétaire de tous les objets qu'il contient.

Il demeure possible de conférer ou révoquer manuellement des privilèges, y compris lorsqu'ils avaient été alloués par ASGARD. Ces modifications ne seront jamais remises en cause par l'extension.

L'idée centrale est de définir trois profils de droits – producteur, éditeur et lecteur.

- ◆ Les **LECTEURS** ont accès en lecture seule aux données du schéma. En pratique, ils ont les droits USAGE sur les schémas et SELECT sur les tables⁵ et séquences.

4 Cette règle est « imposée » au sens où son respect conditionne le bon fonctionnement du système, mais surtout parce qu'ASGARD s'assure qu'elle ne soit jamais enfreinte en procédant dès que besoin à des réaffectations automatiques de propriété. L'utilisateur n'a ainsi jamais à s'en préoccuper. Comme discuté par la suite, cette règle ne concerne cependant que les schémas référencés par ASGARD et rien n'oblige à ce que tous les schémas le soient (sur ce point, cf. notamment VI.7).

5 Y compris vues, vues matérialisées, tables étrangères et tables partitionnées.

- ◆ Les **ÉDITEURS** ont tous les droits des lecteurs, auxquels s'ajoutent les droits qui permettent la modification des données : INSERT, UPDATE et DELETE sur les tables ou assimilés et USAGE sur les séquences.
- ◆ Les **PRODUCTEURS** sont propriétaires des schémas et reçoivent donc automatiquement la propriété de tous les objets qui y sont créés. À ce titre (cf. I.3.A), ils ont tous les droits sur les schémas et les objets qu'ils contiennent. En plus de la visualisation et l'édition des données, les membres du groupe producteur peuvent créer des objets dans le schéma, les supprimer, modifier leur définition ou encore attribuer ou révoquer des droits sur ces objets pour les autres rôles de groupe.

Pour chaque schéma, il est possible de pré-désigner au maximum un rôle de groupe par fonction.

Avec ASGARD, tout schéma a un rôle **PRODUCTEUR** (propriétaire) et au plus un rôle **ÉDITEUR** et un rôle **LECTEUR**.



1 schéma,
3 profils de droits

1 rôle producteur

obligatoire, rôle de groupe



- est propriétaire du schéma et tout ce qu'il contient
- peut créer de nouveaux objets dans le schéma
- peut changer la définition des objets et les supprimer
- peut conférer à d'autres rôles des droits sur les objets
- peut manipuler sans restriction les données des tables et vues
- a plus largement tous les droits sur tous les objets du schéma

0 ou 1 rôle éditeur

optionnel



- accède au schéma
- peut modifier les données des tables et des vues, ajouter et supprimer des enregistrements
- 1.3 • peut utiliser les séquences

0 ou 1 rôle lecteur

optionnel



- accède au schéma
- peut lire les données des tables et des vues
- 1.3 • peut lire la valeur courante d'une séquence

La liste des schémas et des rôles associés est stockée dans la table `gestion_schema` du schéma `z_asgard_admin`. Les utilisateurs la consultent et l'alimentent via la vue

gestion_schema_usr du schéma z_asgard, qui est ainsi l'objet central de la gestion des schémas. Cette vue est présentée en détails dans la partie III.1.

EXEMPLE. Extrait de la vue gestion_schema_usr :

nom_schema character varying	producteur character varying	editeur character varying	lecteur character varying
w_alain	g_snum	g_new_editeur	g_new_lecteur

Dans cet exemple, le schéma de travail w_alain a pour éditeur le groupe g_new_editeur. Les membres de g_new_editeur pourront ainsi modifier les données contenues dans les tables du schéma w_alain, quel que soit le rôle de connexion de l'utilisateur qui, en pratique, aura procédé à la création de l'objet.

II.2 — NOMMAGE

ASGARD n'impose pas de règle pour le nommage des rôles. Il est cependant conseillé d'en adopter une, autant pour la forme des identifiants de connexion attribués aux utilisateurs du serveur que pour celle des rôles de groupe qui permettront de leur conférer des droits.

II.2.A. RÈGLE (OPTIONNELLE) DE NOMMAGE DES RÔLES DE CONNEXION

Il est proposé de retenir la même convention que celle de l'annuaire, soit [prenom]. [nom] dans le cas général⁶, afin de préserver une future possibilité⁷ d'utiliser pour la connexion au serveur PostgreSQL le mot de passe qui sert déjà à l'ouverture de session, la messagerie, les authentifications Cerbère, etc.

BONNE PRATIQUE. Les rôles de connexion individuels sont les identifiants usuels des agents, [prenom].[nom].

Ces noms ont l'inconvénient de contenir des caractères spéciaux (a minima « . »), ce qui oblige à les écrire systématiquement entre guillemets dans les requêtes.

6 Ou l'identifiant de connexion de l'agent, s'il est différent de [prenom].[nom] (noms longs, gestion des homonymies, etc.).

7 Une étude de faisabilité est en cours sur cette question.

EXEMPLE. Requête de création d'un rôle de connexion individuel :

```
CREATE ROLE "jon.snow" LOGIN PASSWORD 'À changer !' ;
```

II.2.B. RÈGLE (OPTIONNELLE) DE NOMMAGE DES RÔLES DE GROUPE

Il est proposé de faire commencer tous les rôles de groupe par `g_` afin de les repérer facilement dans la liste des rôles de connexion/groupe sous pgAdmin.

BONNE PRATIQUE. Les rôles de groupe sont préfixés de `g_`.

Les rôles de groupe d'ASGARD suivent cette règle.

Plus généralement, il est vivement déconseillé de recourir au préfixe `pg_`, déjà utilisé pour les rôles techniques de PostgreSQL (`pg_monitor`, `pg_read_all_settings`...).

II.2.C. CARACTÈRES INTERDITS

ASGARD est permissif sur le nommage des objets et tolère les noms qui ne respectent pas la forme canonique des identifiants PostgreSQL (majuscules, caractères spéciaux, espaces, etc.).

À deux exceptions près :

RÈGLE. Ne jamais utiliser le caractère « = » dans les noms de rôles. Ne jamais utiliser le caractère « % » dans les noms d'objets, quelle que soit leur nature.

D'une manière générale, il reste conseillé d'éviter les caractères spéciaux et tout particulièrement les guillemets simples et doubles, ainsi que les antislashes.

II.3 — RÔLES D'ASGARD

ASGARD crée automatiquement quatre rôles : trois rôles de groupe et un rôle de connexion : `g_consult`, `g_admin`, `g_admin_ext` et `consult.default`.



g_admin

rôle de groupe, créé par ASGARD

- à attribuer à l'ADL et à personne d'autre
 - peut réaliser toutes les actions qui ne requièrent pas un super-utilisateur
 - de nombreuses fonctionnalités d'ASGARD ne sont accessibles qu'à ses membres
 - permission à donner sur tous les rôles de groupe susceptibles d'être utilisés comme producteurs des schémas
- NE PAS MODIFIER LES DROITS DU RÔLE SUR LES OBJETS D'ASGARD

g_consult

rôle de groupe, créé par ASGARD

- à désigner comme lecteur pour tous les schémas contenant des données publiques
 - à attribuer à tous les rôles de connexion
- NE PAS MODIFIER LES DROITS DU RÔLE SUR LES OBJETS D'ASGARD
NE PAS LUI CONFÉRER DE DROITS D'ÉDITION

g_admin_ext

rôle de groupe, créé par ASGARD

- rôle technique
- NE PAS MODIFIER LES DROITS DU RÔLE SUR LES OBJETS D'ASGARD
NE PAS CONFÉRER À D'AUTRES RÔLES QUE G_ADMIN

consult.default

rôle de connexion, créé par ASGARD

- rôle de connexion partagé pour un accès en lecture seule aux données publiques
 - peut être supprimé au gré du service
- NE PAS LUI CONFÉRER D'AUTRES DROITS QUE L'APPARTENANCE À G_CONSULT

éditeur de z_asgard

rôle de groupe, à créer si besoin par le service

- tout utilisateur appelé à gérer les schémas ou les droits doit en être membre (cf. III.1.B)

II.3.A. G_CONSULT — CONSULTATION DES DONNÉES PUBLIQUES

Le rôle de groupe g_consult a pour fonction d'offrir à ses membres un accès en lecture seule à l'ensemble du patrimoine de données du service, à l'exclusion des données confidentielles.

En particulier, il est pré-renseigné comme lecteur pour tous les schémas de la nomenclature nationale (cf. VI.4 pour l'import de ces schémas). Ceci peut être modifié si besoin.

Il est du ressort de l'administrateur de décider pour chaque schéma qu'il crée si g_consult doit ou non en devenir le lecteur, afin d'ouvrir l'accès à son contenu ou le restreindre. En effet, aucun des mécanismes automatisés d'ASGARD ne prévoit l'allocation

automatique de droits à ce rôle s'il n'a pas été explicitement désigné (a priori comme lecteur) dans la table de gestion par l'administrateur⁸.

BONNE PRATIQUE. `g_consult` est désigné comme lecteur de tous les schémas contenant des données publiques.

Sauf cas particulier, il serait ainsi attendu que `g_consult` soit le lecteur de tous les schémas des blocs consultation, référentiel et externe.

BONNE PRATIQUE. `g_consult` étant identifié comme un rôle de lecture, aucun droit en écriture ne lui est octroyé.

Il est notamment vivement déconseillé de désigner `g_consult` comme producteur ou éditeur d'un schéma.

Dans l'absolu, tout utilisateur du serveur est au moins légitime pour lire les données publiques qu'il héberge et devrait donc appartenir à `g_consult`.

BONNE PRATIQUE. Tous les rôles de connexion sont faits membres de `g_consult`.

Lors de l'installation de l'extension, `g_consult` reçoit des droits de lecture sur les vues du schéma `z_asgard`. Ceci doit assurer que tous les rôles qui ont besoin de se référer au contenu de ce schéma (y compris dans le cadre des mécanismes internes d'ASGARD) le puissent.

Ainsi, pour les rôles de connexion des utilisateurs du « menu pour QGIS » (cf. V), l'appartenance à `g_consult` – ou, s'il a été spécifié et qu'il ne s'agit pas de `g_consult`, au rôle lecteur du schéma `z_asgard` – est une obligation. À défaut, le plugin AsgardMenu ne pourra accéder aux données nécessaires à son initialisation et provoquera des erreurs (voire des plantages) au lancement de QGIS.

Il est également indispensable (cf. III.1.A) que tous les membres d'un rôle producteur appartiennent également à `g_consult` – ou au lecteur de `z_asgard` – pour accéder aux vues `gestion_schema_usr` et `gestion_schema_etr`, sans quoi ils ne pourront créer ou modifier la structure d'aucun objet.

⁸ Au contraire de `g_admin`, `g_consult` devient pour ASGARD un rôle de groupe comme les autres une fois l'installation achevée. Il peut être supprimé (mais risque d'être recréé lors des mises à jour de l'extension).

On notera que l'attribution massive de `g_consult` à tous les rôles de connexion du serveur qui n'en seraient pas encore membres peut être réalisée à tout moment en seule une commande, par application de la fonction `asgard_all_login_grant_role`.

II.3.B. G_ADMIN — ADMINISTRATION DE LA BASE

`g_admin` est le groupe des administrateurs.

Ce n'est pas un rôle de connexion (attribut `NOLLOGIN`) et ce n'est pas un super-utilisateur (`NOSUPERUSER`), mais il peut créer, modifier et supprimer des bases (`CREATEDB`), créer, gérer et supprimer des rôles (`CREATEROLE`) et se connecter au serveur en mode réplication (`REPLICATION`). Il a de plus le privilège `CREATE` sur la base, qui lui permet de créer de nouveaux schémas.

Ce rôle est essentiel pour le système de gestion des droits, qui prévoit que lui seul soit habilité à réaliser les actions les plus sensibles et veille à ce qu'il puisse intervenir sur tous les objets de la base, notamment en le rendant automatiquement membre de tous les rôles désignés comme producteurs (hors super-utilisateurs)⁹.

`g_admin` est également le producteur par défaut des schémas de la nomenclature nationale – ce qui peut bien entendu être modifié a posteriori.

Compte-tenu de l'étendue de ses prérogatives, à peine inférieures à celles d'un super-utilisateur, le rôle `g_admin` devrait être strictement réservé aux administrateurs de données localisées (ADL) du service.

BONNE PRATIQUE. Seuls le ou les ADL sont membres de `g_admin`.

Pour le bon fonctionnement d'ASGARD, il est essentiel que le rôle `g_admin` ne soit pas renommé après l'installation et reste quoi qu'il arrive propriétaire du schéma `z_asgard_admin` et de la table `gestion_schema`. Il est cependant possible de changer son nom par un `rechercher/remplacer` dans le script de l'extension préalablement à l'installation, de même d'ailleurs que pour tous les rôles d'ASGARD.

⁹ Aucun contrôle n'est réalisé a posteriori : `g_admin` reste membre du rôle même s'il n'est plus producteur d'aucun schéma et ASGARD n'empêche pas de retirer à `g_admin` des permissions sur certains rôles producteurs. C'est toutefois déconseillé (il est souhaitable que l'administrateur puisse agir sur l'ensemble du patrimoine en cas de problème) et foncièrement inutile, l'attribut `CREATEROLE` autorisant `g_admin` à s'auto-octroyer l'appartenance à tous les rôles hors super-utilisateurs.

RÈGLE. Ne pas modifier les permissions de `g_admin` sur les objets d'ASGARD.

Que `g_admin` dispose des attributs `CREATEROLE` et `CREATEDB` n'interdit pas de doter directement les rôles de connexion des ADL de ces attributs. Cette pratique simplifie considérablement les manipulations courantes sur les rôles (cf. point I.3), notamment celles qui ne sont pas réalisées via la table de gestion d'ASGARD¹⁰.

Comme mentionné précédemment, ASGARD rend systématiquement `g_admin` membre des rôles désignés comme producteurs d'un schéma (lorsqu'il ne l'est pas déjà), et cette opération n'est pas anodine. Automatique, elle ne nécessite aucune intervention de l'utilisateur, cependant elle requiert de celui qui lance la commande définissant le producteur qu'il soit lui-même habilité à accorder cette permission : il doit être membre du dit rôle producteur avec l'option `ADMIN`, disposer de l'attribut `CREATEROLE` ou être membre d'un rôle qui l'a.

Sans cela, ASGARD lui renverra une erreur de ce type¹¹ :

```
ERREUR: ECS0 > TA0 > TA6. Opération interdite. Permissions insuffisantes pour le rôle g_consult.  
DETAIL: GRANT g_consult TO g_admin  
HINT: Votre rôle doit être membre de g_consult avec admin option ou disposer de l'attribut CREATEROLE  
pour réaliser cette opération.
```

C'est notamment lorsqu'un rôle est utilisé pour la première fois comme producteur que la question est susceptible de se poser, du moins si l'opération n'est pas réalisée par un membre de `g_admin` et que ce dernier n'a pas pris la précaution d'attribuer à l'avance le rôle considéré à `g_admin`.

Lorsque les membres de `g_admin` ne sont pas les seuls à assurer la gestion des droits (cf. III.1.B), il est préférable d'anticiper.

BONNE PRATIQUE. Lors de la création d'un rôle de groupe susceptible d'être par la suite identifié comme producteur d'un schéma, rendre immédiatement `g_admin` membre de ce rôle.

10 Car ASGARD saura identifier qu'un rôle de connexion est membre d'un rôle de groupe disposant de `CREATEROLE` et lui faire endosser temporairement ce rôle lorsqu'il lui faut par exemple créer de nouveaux rôles via la table de gestion. Ce n'est pas le cas avec des commandes SQL ordinaires, qui échoueront avec un message d'erreur tel que « ERREUR: droit refusé pour créer un rôle ».

11 La liste de codes en début de message pourra varier selon le contexte.

II.3.C. CONSULT.DEFAULT — RÔLE DE CONNEXION GÉNÉRIQUE

ASGARD crée également un rôle de connexion, `consult.default`.

`consult.default` permet de se connecter à la base avec le mot de passe « `consult.default` ». Ce mot de passe est évidemment modifiable au gré de l'ADL¹².

`consult.default` est membre de `g_consult` et participe au même objectif : rendre largement accessibles les données publiques du serveur (et uniquement celles-là).

BONNE PRATIQUE. `consult.default` ne dispose d'aucun privilège sur la base si ce n'est l'appartenance à `g_consult`.

Autrement dit, `consult.default` n'est pas supposé être membre de rôles de groupe dont les prérogatives seraient plus étendues que celles de `g_consult`. Plus simplement, on pourra considérer qu'il n'a pas à être membre d'un autre rôle que `g_consult`.

BONNE PRATIQUE. Seuls les utilisateurs identifiés par un rôle de connexion personnalisé `[prénom].[nom]` appartiennent à des groupes ayant des droits en écriture ou un accès en lecture à des données à diffusion restreinte.

Rôle de connexion partagé, `consult.default` permet d'éviter la gestion de rôles de connexion individuels pour les utilisateurs qui n'ont besoin que d'un accès en lecture aux schémas qui contiennent des données publiques.

Il peut être supprimé au gré du service, notamment si l'existence d'un tel rôle est contraire à sa politique de sécurité, si la base ne contient aucune information publique et/ou si le nombre d'utilisateurs est suffisamment limité pour attribuer à chacun un rôle de connexion individuel.

À noter que, si par défaut PostgreSQL permet à tous les utilisateurs du serveur de créer de nouveaux objets dans le schéma public, ASGARD supprime cette possibilité évidemment incompatible avec l'existence d'un rôle de connexion générique.

12 Il peut néanmoins rester trivial et être largement diffusé. Ce qui fait la sécurité du système n'est pas sa confidentialité mais le fait qu'un utilisateur se connectant avec `consult.default` ne peut voir que des données publiques et n'a pas la possibilité de les modifier.

II.3.D. G_ADMIN_EXT — RÔLE TECHNIQUE

ASGARD crée enfin le rôle de groupe `g_admin_ext`, dont `g_admin` est et doit rester le seul membre.

`g_admin_ext` est le propriétaire du schéma `z_asgard` et des objets (vues et fonctions) qu'il contient. Il s'agit en quelque sorte d'un alias de `g_admin` avec deux spécificités :

- ◆ il est propriétaire d'objets que nul ne devrait pouvoir altérer hormis l'administrateur ;
- ◆ ses prérogatives sur les objets de `z_asgard_admin` sont limitées au strict nécessaire, car ce sont elles qui déterminent ce que les autres utilisateurs pourront faire via les vues modifiables dont il est propriétaire dans `z_asgard`.

RÈGLE. Ne jamais modifier les permissions de `g_admin_ext` sur les objets d'ASGARD, ni attribuer ce rôle à quiconque.

II.3.E. CAS DE RÔLES PRÉ-EXISTANTS

Si l'un des rôles susmentionnés existait préalablement à l'installation de l'extension, ASGARD lui octroie les droits prévus pour le rôle sur les objets de l'extension, ainsi que (s'il n'en disposait pas déjà) le minimum de permissions génériques nécessaires au bon fonctionnement de l'extension :

- ◆ pour `g_admin` : attribut `CREATEROLE` et privilège `CREATE` sur la base courante, permission sur `g_admin_ext` ;
- ◆ pour `consult.default` : permission sur `g_consult`.
- ◆ pour `g_admin_ext` et `g_consult` : néant.

III — UTILISATION PRATIQUE D'ASGARD

Dans la suite de la documentation, les interactions avec la base de données sont principalement décrites par des commandes en SQL, mais ASGARD permet également de gérer les droits à l'aide des fonctionnalités de l'interface de pgAdmin, ou encore depuis QGIS (cf. VII).

III.1 — VUE GESTION_SCHEMA_USR

Concrètement, toutes les données d'ASGARD se trouvent dans la table `gestion_schema` du schéma `z_asgard_admin`.

Cette table est appelée à répertorier tous les schémas existants dans la base de données (hors schémas système), ainsi que des schémas pré-définis en vue d'une création future. En particulier, elle peut être initialisée avec les schémas de la nomenclature nationale (cf. VI.4), qui sont pour l'heure uniquement des schémas de consultation `c_`. Ces schémas appartenant à la nomenclature nationale sont protégés de certaines modifications¹³.

La table `gestion_schema` n'a toutefois pas vocation à être consultée ou modifiée directement. Seuls les membres de `g_admin` auraient les privilèges suffisant pour le faire et, même pour eux, ce n'est pas recommandé. Le point d'entrée pour utiliser ASGARD est la vue `gestion_schema_usr` dans le schéma `z_asgard`.

¹³ Cf. description du champ nomenclature ci-après.



tables et vues d'ASGARD

◆ z_asgard_admin

réservé à l'ADL (membres de g_admin)

■ gestion_schema

- table de stockage des données d'ASGARD
- alimente les vues de z_asgard
- NE PAS UTILISER DIRECTEMENT

■ asgard_parametre

- table de paramétrage d'ASGARD
- à l'usage de l'ADL
- à ce stade, sert uniquement pour le plugin adapté de MenuBuilder (cf. V)

◆ z_asgard

accessible à tous les utilisateurs
(membres de g_consult)

■ gestion_schema_usr

VUE UTILISATEUR

- outil de gestion des schémas et des droits
- à l'usage de l'ADL, des administrateurs délégués et des producteurs des schémas

■ gestion_schema_etr

- vue technique, sans intérêt pour les utilisateurs

■ qgis_menubuilder_metadata

- vue du plugin adapté de MenuBuilder (cf. V)
- répertorie les tables et vues auxquelles un utilisateur a accès

ASGARD intercepte en outre toutes les commandes SQL qui sont pertinentes pour la gestion des droits : CREATE SCHEMA, DROP SCHEMA, ALTER SCHEMA OWNER/RENAME TO, CREATE [OBJET] et certains ALTER [OBJET]. S'il y a lieu, la table de gestion est mise à jour en conséquence.

Avec ASGARD, la gestion des droits se fait soit par les commandes habituelles, soit via la vue `gestion_schema_usr` du schéma `z_asgard`.

Dans la suite du document, on désignera l'espace de stockage des données d'ASGARD sous le terme générique de « table de gestion », sans rappeler systématiquement que l'utilisateur y accède par l'intermédiaire d'une vue.

III.1.A. ACCÈS À LA VUE

La vue `gestion_schema_usr` se présente sous la forme suivante :

nom_schema character varying	bloc character varying (1)	nomenclature boolean	niv1 character varying	niv1_abr character varying	niv2 character varying	niv2_abr character varying	creation boolean	producteur character varying	editeur character varying	lecteur character varying
z_asgard	z	false	ASGARD	[null]	[null]	[null]	true	g_admin_ext	g_createschema	g_consult
z_asgard_admin	z	false	ASGARD	[null]	RESERVE ADL	[null]	true	g_admin	[null]	[null]
c_agri_agroalim...	c	true	agriculture	agri	agro_alimentaire	agroalimentaire	false	g_admin	[null]	g_consult
c_agri_environn...	c	true	agriculture	agri	agri_environne...	environnement	false	g_admin	[null]	g_consult
c_agri_exploi_el...	c	true	agriculture	agri	exploitation_ele...	exploit_elevage	false	g_admin	[null]	g_consult
c_agri_parcellair...	c	true	agriculture	agri	parcellaire_agri...	parcellaire_agri...	false	g_admin	[null]	g_consult
c_agri_sante_an...	c	true	agriculture	agri	sante_animale	sante_animale	false	g_admin	[null]	g_consult

Elle est accessible en lecture aux membres de `g_consult`, soit en principe tous les rôles de connexion (cf. II.3.A). Cela ne signifie toutefois pas que tous les utilisateurs du serveur pourront visualiser à travers elle l'ensemble des informations contenues dans la table `gestion_schema`.

En premier lieu, la vue `gestion_schema_usr` ne reprend pas tous les champs de `gestion_schema`, mais seulement ceux qui ont vocation à être alimentés manuellement.

En second lieu, `gestion_schema_usr` est définie de manière à n'afficher que les lignes sur lesquelles l'utilisateur est légitime à intervenir¹⁴ :

- ◆ les schémas existants si et seulement s'il est membre de leur rôle producteur ;
- ◆ les schémas pré-définis (non créés) si et seulement si l'utilisateur est lui-même habilité à créer des schémas ou s'il a explicitement endossé¹⁵ le rôle désigné comme futur producteur.

Les membres de `g_admin` voient quoi qu'il arrive tous les enregistrements de la table de gestion.

gestion_schema_usr ne montre à l'utilisateur que les schémas dont il est habilité à gérer les droits.

Par ailleurs, tous les utilisateurs membres des groupes producteurs ont besoin d'un accès en lecture à `z_asgard` et son contenu pour pouvoir créer des objets ou modifier leur définition. Autrement dit, il leur faut être membre du rôle lecteur de `z_asgard` et/ou de

¹⁴ À noter que, si un utilisateur lambda souhaite obtenir des informations sur les objets existants, les propriétaires et les droits, elles sont accessibles à tous dans les schémas `pg_catalog` et `information_schema`.

¹⁵ Par exemple par une commande `SET ROLE [producteur]`, ce qui requiert bien entendu que l'utilisateur soit membre dudit rôle producteur.

`g_consult`, qui dispose nativement de ces droits. Ceci rejoint le conseil précédemment formulé de rendre systématiquement tous les rôles de connexion membres de `g_consult` (cf. II.3.A).

RÈGLE. Tous les membres de rôles identifiés comme producteurs doivent être membres de `g_consult`.

Une fonction, `asgard_all_login_grant_role`, est mise à disposition pour attribuer massivement et rapidement un rôle (ici `g_consult`) à tous les rôles de connexion du serveur.

III.1.B. DÉLÉGUER LA GESTION DES DROITS

Par défaut, `g_admin` et ses membres sont les seuls à pouvoir éditer `gestion_schema_usr`, et à travers elle la table `gestion_schema`.

Si l'ADL souhaite déléguer à d'autres utilisateurs la possibilité de gérer les droits et les schémas, il lui faut désigner un rôle **éditeur** pour le schéma `z_asgard`¹⁶ et s'assurer que tous les utilisateurs concernés soient membres de ce rôle.

EXEMPLE. Pour déclarer le rôle de groupe `g_admin_delegate` en tant qu'éditeur de `z_asgard` avec une commande SQL :

```
UPDATE z_asgard.gestion_schema_usr
SET editeur = 'g_admin_delegate'
WHERE nom_schema = 'z_asgard' ;
```

Différents types d'actions peuvent alors être réalisés via la table de gestion et, dans certains cas, PostgreSQL requiert des droits supplémentaires. Ceux-ci sont toujours acquis pour les membres de `g_admin`, mais pas nécessairement pour les membres de l'éditeur de `z_asgard` :

- ◆ pour créer de nouveaux schémas, renommer un schéma ou supprimer un schéma, il faut en plus le privilège `CREATE` sur la base¹⁷ ;
- ◆ pour désigner comme producteur, éditeur ou lecteur des rôles qui n'existent pas encore – et donc les créer – il faut en plus l'attribut `CREATEROLE` ;

¹⁶ Ce qui implique que le schéma `z_asgard` ait été préalablement référencé dans la table de gestion. Cf. VI.4.

¹⁷ En SQL : `GRANT CREATE ON DATABASE [base] TO [rôle]`. Ce privilège sera de préférence attribué à un rôle de groupe dont on rendra membre les rôles de connexion concernés, comme dans l'exemple qui conclut cette partie.

- ◆ pour désigner comme producteur un rôle dont `g_admin` n'est pas encore membre, il faut en plus l'attribut `CREATEROLE` ou être membre du rôle considéré avec l'option `ADMIN`¹⁸ ;
- ◆ pour désigner comme producteur un rôle dont on n'est pas membre soi-même, il faut en plus l'attribut `CREATEROLE`.

Cf. I.3.C pour plus de précisions sur l'attribut `CREATEROLE`.

Il revient à l'ADL d'attribuer les privilèges susmentionnés à qui de droit. Si un utilisateur tente de réaliser une action pour laquelle ses droits sont insuffisants, ASGARD lui renvoie un message d'erreur explicitant les permissions manquantes.

Réciproquement, il importe de noter que si un rôle de groupe autre que l'ADL dispose du privilège `CREATE` sur la base, ses membres devront aussi appartenir à l'éditeur du schéma `z_asgard`, sans quoi leurs requêtes de création, modification ou suppression de schémas se solderont par des erreurs.

EXEMPLE. Tentative infructueuse de modification du nom d'un schéma par un rôle disposant du privilège `CREATE` sur la base et membre du rôle producteur du schéma, mais qui n'est pas habilité à modifier la table de gestion d'ASGARD.

```
ALTER SCHEMA w_snow RENAME TO w_jsnow ;
```

ERREUR: ECS0 > ECS2. Vous devez être membre du groupe éditeur du schéma `z_asgard` pour réaliser cette opération.

RÈGLE. Pour déléguer la création de schémas à un rôle, il faut lui conférer le privilège `CREATE` sur la base et s'assurer qu'il soit membre de l'éditeur du schéma `z_asgard`.

EXEMPLE. `g_admin_delegue` a été défini comme éditeur pour `z_asgard`. On décide maintenant que certains utilisateurs membres de `g_admin_delegue` devront pouvoir créer des schémas et modifier ou supprimer ceux dont ils sont propriétaires. Ces utilisateurs seront membres du rôle `g_createschema`, auquel on donne le privilège `CREATE` sur la base.

En SQL :

```
GRANT CREATE ON DATABASE geobase_snum TO g_createschema ;
```

¹⁸ Sur cette question, se référer aux derniers paragraphes de la partie II.3.B.

... ou, dans pgAdmin (propriétés de la base) :

Bénéficiaire	Droits
g_createschema	<input type="checkbox"/> ALL <input checked="" type="checkbox"/> CREATE <input type="checkbox"/> TEMPORARY <input type="checkbox"/> CONNECT <input type="checkbox"/> WITH GRANT OPTION <input type="checkbox"/> WITH GRANT OPTION <input type="checkbox"/> WITH GRANT OPTION <input type="checkbox"/> WITH GRANT OPTION
g_admin	CTc
PUBLIC	Tc

Pour accorder à jon.snow le droit de créer des schémas :

```
GRANT g_admin_createschema TO "jon.snow" ;
GRANT g_admin_delegate TO "jon.snow" ;
```

... ou, dans pgAdmin :

Rôle de connexion - jon.snow

General Définition Droits **Appartenance** Paramètres

Rôles

- g_admin_delegate
- g_consult
- g_createschema

Cocher la case pour considérer les rôles WITH ADMIN OPTION.

On pourrait aussi conférer à tous les membres actuels et futurs de g_createschema l'appartenance à g_admin_delegate par la commande :

```
GRANT g_admin_delegate TO g_createschema ;
```

L'ajout de nouveaux enregistrements dans gestion_schema_usr est réservé aux utilisateurs habilités à créer des schémas (privilège CREATE sur la base), y compris pour des schémas non créés.

III.1.C. CAS PARTICULIER D'UN SCHÉMA DONT LE PRODUCTEUR EST UN SUPER-UTILISATEUR

Par construction, `g_admin` et ses membres voient dans `gestion_schema_usr` tous les schémas référencés par ASGARD et, dans le cas général, ASGARD fait en sorte qu'ils soient habilités à intervenir sur ces schémas en rendant automatiquement `g_admin` membre de leur rôle producteur (cf. II.3.B).

Cette règle souffre toutefois une exception : jamais ASGARD ne tentera d'attribuer à `g_admin` de permission sur un rôle super-utilisateur. Ainsi, si des rôles super-utilisateurs – par exemple `postgres` ou, pour les distributions EOLE, `ad1` – sont propriétaires de schémas, eux seuls pourront les administrer via ASGARD. Bien qu'il les voie dans la table de gestion (sous réserve qu'ils aient été référencés), `g_admin` ne pourra réaliser que des actions limitées sur ces schémas :

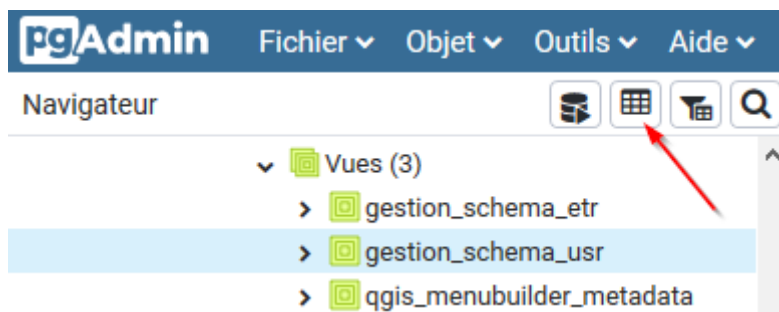
- ◆ modifier les champs nomenclature, `niv1`, `niv2`, `niv1_abr` et `niv2_abr` ;
- ◆ mettre le schéma à la corbeille.

`g_admin` n'est pas non plus habilité à référencer lui-même dans ASGARD un schéma dont un super-utilisateur est propriétaire (cf. VI.4) ou à créer des schémas en désignant comme producteur un super-utilisateur sur lequel il n'a aucune permission. Il peut par contre pré-référencer dans la table de gestion d'ASGARD des schémas inactifs avec comme futur producteur un super-utilisateur (cf. III.4), qui pourront être activés ultérieurement par la super-utilisateur en question.

Sauf raison spécifique, il restera préférable d'attribuer à `g_admin` et non à un super-utilisateur les schémas sur lesquels l'ADL est le seul à être légitime pour agir.

III.1.D. PGADMIN ET L'ÉDITION DES VUES

D'une manière générale, pgAdmin4 propose une vue tabulaire de consultation et édition des données, accessible en choisissant Afficher/éditer les données après un clic droit sur une table, vue ou vue matérialisée dans l'explorateur, ou en cliquant sur l'icône dédiée de la barre d'outils.



Pour l'heure, cette fonctionnalité permet de visualiser les données d'une vue, mais pas de les éditer. Ainsi il n'existe actuellement qu'une méthode pour modifier le contenu de la vue `gestion_schema_usr` dans pgAdmin4 : lancer des commandes UPDATE, INSERT, DELETE dans l'éditeur de requête.

La vue `gestion_schema_usr` peut par contre être ouverte et modifiée dans QGIS.

III.2 — CHAMPS DE LA TABLE DE GESTION

III.2.A. BLOC

`bloc` est une chaîne de caractères de longueur 1.

Il indique le bloc fonctionnel du schéma suivant les conventions de nommage de la nomenclature nationale :

- ◆ `c` : schémas de consultation (mise à disposition de données publiques) ;
- ◆ `w` : schémas de travail ou d'unité ;
- ◆ `s` : géostandards ;
- ◆ `p` : schémas thématiques ou dédiés à une application ;
- ◆ `r` : référentiels ;
- ◆ `x` : données confidentielles ;
- ◆ `e` : données externes (open data, etc.) ;
- ◆ `z` : utilitaires.

La valeur « `d` » joue un rôle spécifique. Elle indique que le schéma est mis à la « corbeille » et sera dès lors automatiquement supprimé, ainsi que son contenu, si `creation` est mis à `False`¹⁹.

Pour supprimer un schéma via la table de gestion, il faut d'abord changer son bloc en « `d` » (corbeille) puis basculer `creation` sur `False`.

¹⁹ Sur le sujet de la suppression des schémas, cf. aussi III.5.

bloc peut-être NULL si le schéma n'est pas défini comme appartenant à l'un des blocs fonctionnels.

Il est également possible d'utiliser des blocs hors nomenclature (par exemple « a »).

Par principe, dès lors qu'un bloc est renseigné, le nom du schéma (champ `nom_schema`) commence par un préfixe formé de la lettre du bloc suivie d'un tiret bas, `[bloc]_`.

EXEMPLE. `w_jon_snow` est un schéma de travail de l'utilisateur `jon.snow`. Son champ `bloc` vaut `'w'`.

Cette règle connaît une seule exception : les schémas à la corbeille conservent leur préfixe d'origine pour faciliter la restauration.

ASGARD assure automatiquement la cohérence entre le bloc et le préfixe du nom du schéma lorsque l'utilisateur agit sur l'un ou l'autre.

Le mécanisme de mise en cohérence suit deux principes : en cas de modification, la nouvelle saisie prévaut toujours sur les informations pré-existantes et, si l'utilisateur saisit simultanément des informations contradictoires dans `bloc` et `nom_schema`, le bloc prévaut sur le préfixe de `nom_schema` tant qu'il n'est pas vide.

Les différents cas de figure sont détaillés ci-après.

À la création d'un schéma (hors corbeille) :

- ◆ si le bloc n'est pas renseigné mais que le nom du schéma comporte un préfixe, le bloc est mis à jour selon ce préfixe ;

EXEMPLE. Si le schéma `w_jon_snow` est créé par une commande `CREATE SCHEMA`, qui ne permet pas de spécifier un bloc, ASGARD saura interpréter le préfixe « `w_` » dans le nom du schéma et indiquera tout de même `'w'` dans le champ `bloc` de la table de gestion.

utilisateur → NULL + `w_jon_snow`
arrivée : `w` + `w_jon_snow`

- ◆ si le nom du schéma ne comporte pas de préfixe mais que le bloc est renseigné, un préfixe correspondant au bloc est ajouté au nom du schéma ;

EXEMPLE. Si l'utilisateur enregistre dans la table de gestion un schéma `bibliotheque` avec un bloc valant `'c'`, le schéma sera renommé en `c_bibliotheque` par ASGARD.

utilisateur → `c` + `bibliotheque`

arrivée : c + c_bibliotheque

- ◆ si le nom du schéma comporte un préfixe qui ne correspond pas au bloc renseigné, le préfixe est modifié selon le bloc.

EXEMPLE. Si, par mégarde, l'utilisateur enregistre dans la table de gestion un schéma b_bibliotheque avec un bloc valant 'c', le schéma sera automatiquement renommé en c_bibliotheque.

utilisateur → c + b_bibliotheque
arrivée : c + c_bibliotheque

Lors d'une modification (hors corbeille) :

- ◆ si le bloc change, le préfixe du schéma est mis à jour en conséquence. Cette règle vaut également si le préfixe avait lui-même été modifié en parallèle ;

EXEMPLE. Si le schéma w_bibliotheque, appartenant initialement au bloc des schémas de travail (bloc = 'w'), est déclaré comme étant un schéma de consultation (bloc = 'c'), son nom est automatiquement transformé en c_bibliotheque.

origine : w + w_bibliotheque
utilisateur → c + w_bibliotheque
arrivée : c + c_bibliotheque

EXEMPLE. Si le schéma w_bibliotheque, appartenant initialement au bloc des schémas de travail (bloc = 'w'), est déclaré comme étant un schéma de consultation (bloc = 'c') et simultanément renommé en l_librairie (préfixe non cohérent avec le nouveau bloc), son nom est automatiquement transformé en c_librairie.

origine : w + w_bibliotheque
utilisateur → c + l_librairie
arrivée : c + c_librairie

- ◆ si le bloc est supprimé tandis que le préfixe du schéma reste inchangé, ASGARD supprime de lui-même le préfixe du schéma ;

EXEMPLE. Si l'utilisateur met à NULL le champ bloc pour le schéma c_bibliotheque, ce dernier sera renommé en bibliotheque.

origine : c + c_bibliotheque
utilisateur → NULL + c_bibliotheque
arrivée : NULL + bibliotheque

- ◆ si le bloc est supprimé et que le préfixe du schéma change, ASGARD met à jour le bloc selon le nouveau préfixe du schéma (comme il l'aurait fait si le bloc n'avait pas été modifié) ;

EXEMPLE. Si l'utilisateur met à NULL le champ bloc pour le schéma c_bibliotheque, qu'il renomme parallèlement en w_librairie, ASGARD conserve le nom w_librairie et inscrira 'w' dans le champ bloc. À noter que le résultat aurait été le même si bloc n'avait pas été vidé.

origine : c + c_bibliotheque
utilisateur → NULL + w_librairie
arrivée : w + w_librairie

- ◆ si le préfixe dans le nom du schéma change sans que le bloc ne soit modifié, ce dernier est automatiquement corrigé en conséquence ;

EXEMPLE. Si l'utilisateur lance la commande suivante :

```
ALTER SCHEMA w_bibliotheque RENAME TO c_bibliotheque ;
```

... le bloc est automatique changé de 'w' à 'c'.

origine : w + w_bibliotheque
utilisateur → w + c_bibliotheque
arrivée : c + c_bibliotheque

- ◆ si le préfixe dans le nom du schéma est supprimé tandis que le bloc reste inchangé, ASGARD met à NULL le champ bloc ;

EXEMPLE. Si l'utilisateur lance la commande suivante :

```
ALTER SCHEMA w_bibliotheque RENAME TO bibliotheque ;
```

... le bloc est automatique changé de 'w' à NULL.

origine : w + w_bibliotheque
utilisateur → w + bibliotheque
arrivée : NULL + bibliotheque

- ◆ si le préfixe dans le nom du schéma est supprimé tandis que le bloc est modifié, ASGARD met à jour le nom du schéma selon le nouveau bloc.

EXEMPLE. Si l'utilisateur passe le schéma w_bibliotheque du bloc « w » au bloc « c », tout en le renommant en librairie (pas de préfixe), ASGARD ajoutera automatique-

ment le préfixe correspondant au nouveau bloc, ce qui donnera un schéma `c_li-
brairie`.

origine : `w + w_bibliotheque`
utilisateur → `c + librairie`
arrivée : `c + c_librairie`

Avec la corbeille, l'idée à retenir est qu'ASGARD fait en sorte qu'un schéma de la corbeille ait toujours pour bloc « d » et jamais comme préfixe « d_ » :

- ◆ lorsqu'un schéma est mis à la corbeille sans modification de son préfixe, ledit préfixe est conservé. Ceci vaut aussi pour un schéma sans bloc qui n'avait pas de préfixe ;

EXEMPLE. Lorsque le schéma `c_bibliotheque`, appartenant initialement au bloc « c », est mis à la corbeille par bascule du bloc à « d », son nom reste `c_bibliotheque`.

origine : `c + c_bibliotheque`
utilisateur → `d + c_bibliotheque`
arrivée : `d + c_bibliotheque`

- ◆ lorsque le préfixe est modifié simultanément à la mise à la corbeille et que le nouveau préfixe n'est pas « d_ », le nouveau préfixe est conservé. Ceci vaut également pour une suppression de préfixe au moment de la mise à la corbeille ;

EXEMPLE. Si le schéma `c_bibliotheque` est mis à la corbeille sous le nom `w_bibliotheque`, il s'appellera `w_bibliotheque` une fois dans la corbeille.

origine : `c + c_bibliotheque`
utilisateur → `d + w_bibliotheque`
arrivée : `d + w_bibliotheque`

- ◆ si l'utilisateur tente de mettre un préfixe « d_ » sur un schéma qui se trouve déjà dans la corbeille, ASGARD rétablit le préfixe antérieur (ou l'absence de préfixe, le cas échéant) ;

EXEMPLE. Toutes les tentatives pour imposer un préfixe « d_ » au schéma `c_bibliotheque` de la corbeille seront annulées par ASGARD.

origine : `d + c_bibliotheque`
utilisateur → `d + d_bibliotheque`
arrivée : `d + c_bibliotheque`

- ◆ si l'utilisateur met un préfixe « d_ » sur un schéma qui n'est pas encore dans la corbeille, sans modifier le bloc lui-même ou en le mettant à 'd' ou NULL²⁰, alors le schéma est mis à la corbeille – son bloc devient 'd' – et le préfixe antérieur (ou l'absence de préfixe) est rétabli ;

EXEMPLE. Si le schéma `c_bibliotheque` est mis à la corbeille par la commande suivante :

```
ALTER SCHEMA c_bibliotheque RENAME TO d_bibliotheque ;
```

... alors le schéma part bien dans la corbeille (son bloc devient 'd') et il retrouve son préfixe d'origine « c_ ».

origine : c + `c_bibliotheque`
utilisateur → c + `d_bibliotheque`
arrivée : d + `c_bibliotheque`

- ◆ modifier le préfixe d'un schéma de la corbeille n'a aucun effet sur le bloc, qui reste à 'd' quoi qu'il arrive. Autrement dit, pour sortir un schéma de la corbeille, il faut changer son bloc, pas son nom.

EXEMPLE. Si l'utilisateur renomme le schéma `c_bibliotheque` de la corbeille en `w_bibliotheque`, le schéma sera effectivement renommé, mais demeure dans la corbeille.

origine : d + `c_bibliotheque`
utilisateur → d + `w_bibliotheque`
arrivée : d + `w_bibliotheque`

III.2.B. NOMENCLATURE

`nomenclature` est un booléen.

Il indique si un schéma appartient à la nomenclature nationale.

Ne peut être NULL, valeur par défaut `False`.

Les enregistrements correspondant aux schémas de la nomenclature nationale ne peuvent pas être effacés de la table. Ceci empêche également le déréférencement d'un schéma dont le champ `nomenclature` vaut `True` (cf. IV.6).

20 Si le bloc est modifié et que la nouvelle valeur n'est ni 'd' ni NULL, alors la modification du bloc prévaudra sur celle du nom et le schéma n'ira pas dans la corbeille.

Les membres de `g_admin` peuvent basculer ce champ de `True` à `False` et faire ainsi sortir un schéma de la nomenclature nationale. Ils sont également les seuls à pouvoir identifier un schéma comme appartenant à la nomenclature en mettant le champ sur `True`.

Toute modification des champs `bloc`, `niv1`, `niv1_abr`, `niv2`, `niv2_abr` et `nom_schema` est interdite pour les schémas de la nomenclature si elle n'est pas réalisée par un membre du groupe `g_admin`.

III.2.C. NIV1

`niv1` est une chaîne de caractères.

Il donne le « domaine » auquel appartient le schéma, soit le premier niveau de l'arborescence de classement. Ce champ est exploité pour présenter aux utilisateurs de QGIS un menu hiérarchique d'accès au patrimoine de données (cf. V.1). S'il n'est pas renseigné, c'est le nom du schéma qui sera utilisé dans ce menu.

Ce champ peut contenir des majuscules, des caractères spéciaux et des espaces simples (pas de tabulation ni de retour à la ligne).

III.2.D. NIV1_ABR

`niv1_abr` est une chaîne de caractères.

Pour les schémas de la nomenclature nationale, il correspond à la forme normalisée de `niv1` qui était utilisée dans l'arborescence COVADIS de la Géobase et respecte les règles de nommage de PostgreSQL (pas d'espace, pas de majuscule, pas de caractère spécial...).

Pour l'heure, ce champ n'est pas exploité par ASGARD.

III.2.E. NIV2

`niv2` est une chaîne de caractères.

Il donne le « sous-domaine » auquel appartient le schéma, soit le second niveau de l'arborescence de classement. Ce champ est exploité pour présenter aux utilisateurs de QGIS un menu hiérarchique d'accès au patrimoine de données (cf. V.1). S'il n'est pas renseigné, les données apparaissent directement sous le premier niveau.

Ce champ peut contenir des majuscules, des caractères spéciaux et des espaces simples (pas de tabulation ni de retour à la ligne).

III.2.F. NIV2_ABR

`niv2_abr` est une chaîne de caractères.

Pour les schémas de la nomenclature nationale, il correspond à la forme normalisée de niv2 qui était utilisée dans l'arborescence COVADIS de la Géobase et respecte les règles de nommage de PostgreSQL (pas d'espace, pas de majuscule, pas de caractère spécial...).

Pour l'heure, ce champ n'est pas exploité par ASGARD.

III.2.G. NOM_SCHEMA

nom_schema est une chaîne de caractères.

Il donne le nom du schéma. C'est la clef primaire de la table de gestion, il ne peut être NULL et doit être unique²¹.

Lorsque creation vaut True, il s'agit nécessairement du nom d'un schéma existant dans la base. Si le schéma est renommé, nom_schema est automatiquement mis à jour par ASGARD (cf. III.6.A).

III.2.H. CREATION

creation est un booléen.

Il rend compte de l'état d'existence du schéma (créé ou non).

Ne peut être NULL, valeur par défaut False.

Si un administrateur²² bascule creation de False à True pour un ou plusieurs schémas, cela déclenche automatiquement la création du ou des schéma-s, en tenant compte de leurs producteurs pré-définis et, le cas échéant, des éditeurs et lecteur renseignés. Cf. III.3 pour plus de détails sur cette fonctionnalité.

Si creation vaut True, il ne peut être basculé manuellement à False que si le bloc est d (corbeille). Une suppression du schéma via une commande SQL DROP SCHEMA²³ est interceptée par ASGARD, qui bascule le champ à False même si le schéma n'était pas dans la corbeille.

III.2.I. PRODUCTEUR

producteur est une chaîne de caractères.

21 L'unicité portant sur l'ensemble des schémas de la table de gestion, qu'ils soient créés ou non.

22 Ou un autre utilisateur habilité. Cf. III.1.B.

23 Ou les fonctionnalités des explorateurs de pgAdmin, QGIS et DBManager prévues à cet effet.

Il désigne le rôle de groupe du profil « producteur » pour le schéma, c'est-à-dire le rôle qui en est le propriétaire et a tous les droits sur le schéma et les futurs objets qu'il contiendra.

Ne peut être NULL²⁴, pré-renseigné avec `g_admin` pour les schémas de la nomenclature nationale. Si le rôle indiqué n'existe pas et que `creation` vaut `True`, il est créé – sous réserve toutefois que l'opérateur soit habilité à créer des rôles (cf. III.1.B).

RÈGLE. Le producteur ne peut pas être un rôle de connexion, sauf s'il s'agit d'un super-utilisateur.

Après modification du producteur (pour un schéma existant), l'ancien producteur n'aura plus aucun privilège sur le schéma et les objets qu'il contient²⁵. Le nouveau producteur sera propriétaire du schéma et de tous ses objets, et il aura sur eux très précisément les privilèges qu'avait auparavant l'ancien. Concrètement, cela signifie que si un droit avait fait l'objet d'une révocation manuelle pour l'ancien producteur (par exemple pour éviter une modification accidentelle), il restera révoqué pour le nouveau. En tant que propriétaire des objets, celui-ci pourra bien entendu le rétablir s'il le souhaite.

Ce principe de transfert des privilèges correspond au fonctionnement naturel de PostgreSQL en cas de changement de propriétaire. ASGARD le conserve pour les producteurs et le reproduit pour les éditeurs et lecteurs (cf. III.2.J et III.2.K).

Dans le contexte d'ASGARD, lancer une commande de modification du propriétaire d'un schéma revient à désigner un nouveau producteur. Sur ce sujet, cf. III.6.B.

ASGARD impose aux champs producteur, editeur et lecteur de prendre des valeurs différentes (ou NULL pour les deux derniers).

Au contraire de `nom_schema`, le champ producteur n'est pas immédiatement mis à jour lorsqu'un rôle est renommé. Dans la mesure où ASGARD conserve également l'identifiant système du rôle producteur de tous les schémas créés, il saura néanmoins toujours de quel rôle il était question. Pour plus de précisions, cf. III.10.

III.2.J. EDITEUR

`editeur` est une chaîne de caractères.

24 Ce champ n'a pas de valeur par défaut au sens de PostgreSQL, cependant les mécanismes internes d'ASGARD font que le rôle `g_admin` sera utilisé en l'absence d'autre valeur.

25 Du moins aucun qui lui soit directement attribué. Le cas échéant, il bénéficie toujours des privilèges conférés à tous les utilisateurs via le pseudo-rôle public et des privilèges des rôles dont il est membre.

Il désigne le rôle de groupe du profil « éditeur » pour le schéma, qui est habilité à modifier les données contenues dans les futures tables du schéma, mais pas à agir sur les objets eux-mêmes, que ce soit pour les supprimer, modifier leur structure ou en créer de nouveaux.

Peut être NULL.

Le pseudo-rôle public, qui représente l'ensemble des utilisateurs du serveur, peut être désigné comme éditeur. Il est également possible de choisir un rôle de connexion, bien que cela ne soit pas recommandé.

Si le rôle indiqué n'existe pas et que `creation` vaut `True`, il est créé – sous réserve toutefois que l'opérateur soit habilité à créer des rôles (cf. III.1.B).

Après modification de l'éditeur (pour un schéma existant), l'ancien éditeur n'aura plus aucun privilège direct sur le schéma et les objets qu'il contient. Le nouvel éditeur aura sur eux très précisément les privilèges qu'avait auparavant l'ancien, qui peuvent différer des droits standards si des modifications manuelles ont été réalisées, que ce soit pour révoquer des privilèges ou en attribuer de nouveaux.

Il est évidemment possible de vider le champ sans désigner de nouvel éditeur.

éditeur n'est pas immédiatement mis à jour lorsqu'un rôle est renommé ou supprimé. Dans la mesure où ASGARD conserve également l'identifiant système du rôle éditeur de tous les schémas créés (lorsqu'il est défini), il saura néanmoins toujours de quel rôle il était question. Pour plus de précisions, cf. III.10.

ASGARD impose aux champs producteur, éditeur et lecteur de prendre des valeurs différentes (ou NULL pour les deux derniers).

III.2.K. LECTEUR

Lecteur est une chaîne de caractères.

Il désigne le rôle de groupe qui a accès en lecture à tous les futurs objets du schéma.

Peut être NULL.

Le pseudo-rôle public, qui représente l'ensemble des utilisateurs du serveur, peut être désigné comme lecteur. Il est également possible de choisir un rôle de connexion, bien que cela ne soit pas recommandé.

Si le rôle indiqué n'existe pas et que `creation` vaut `True`, il est créé – sous réserve toutefois que l'opérateur soit habilité à créer des rôles (cf. III.1.B).

Après modification du lecteur (pour un schéma existant), l'ancien lecteur n'aura plus aucun privilège direct sur le schéma et les objets qu'il contient. Le nouveau lecteur aura sur eux très précisément les privilèges qu'avait auparavant l'ancien, qui peuvent différer des

droits standards si des modifications manuelles ont été réalisées, que ce soit pour révoquer des privilèges ou en attribuer de nouveaux.

Il est évidemment possible de vider le champ sans désigner de nouveau lecteur.

lecteur n'est pas immédiatement mis à jour lorsqu'un rôle est renommé ou supprimé. Dans la mesure où ASGARD conserve également l'identifiant système du rôle lecteur de tous les schémas créés (lorsqu'il est défini), il saura néanmoins toujours de quel rôle il était question. Pour plus de précisions, cf. III.10.

ASGARD impose aux champs producteur, editeur et lecteur de prendre des valeurs différentes (ou NULL pour les deux derniers).

III.3 — CRÉATION DE SCHÉMAS DE LA NOMENCLATURE NATIONALE

Un des avantages d'ASGARD est de permettre l'import de la nomenclature nationale dans sa table de gestion, ce qui représente à ce jour 108 schémas du bloc « consultation ». Par défaut, leur producteur est `g_admin` – seuls les administrateurs peuvent ajouter des objets dans ces schémas – et leur lecteur `g_consult` – tous les membres de `g_consult`, donc en principe tous les utilisateurs, ont accès à ces schémas, y compris le rôle de connexion générique `consult.default`. Ces valeurs sont pleinement modifiables.

Pour travailler sur les schémas de la nomenclature nationale, il faut d'abord les intégrer dans la table de gestion (cf. VI.4).

Par la suite, pour créer effectivement des schémas appartenant à cette nomenclature, il suffit de basculer leur champ `creation` à `True` dans la vue `gestion_schema_usr`.

```
UPDATE z_asgard.gestion_schema_usr
SET creation = True
WHERE nom_schema = 'c_air_clim_changement' ;
```

Il est également possible de les créer par une commande SQL.

```
CREATE SCHEMA c_air_clim_changement ;
```

ASGARD intercepte la commande et, repérant que le schéma qui vient d'être créé est enregistré dans la table de gestion, met à jour l'enregistrement correspondant – `creation` passe à `True`. Les privilèges du schéma seront en retour modifiés selon les prescriptions de la table de gestion pour les fonctions d'éditeur et de lecteur. Le producteur est pour sa part mis à jour dans la table de gestion en fonction du propriétaire du schéma qui vient d'être créé, c'est-à-dire l'utilisateur courant ou le rôle spécifié par la clause `AUTHORIZATION`.

On rappelle qu'ASGARD interdit qu'un rôle de connexion soit producteur.

EXEMPLE. Le rôle de connexion `jon.snow` dispose des permissions nécessaires pour créer des schémas (cf. III.1.B). Mais s'il lance une commande `CREATE SCHEMA` sans plus de précaution, ASGARD lui retourne une erreur :

```
CREATE SCHEMA c_air_clim_changement ;
```

ERREUR: ECS0 > TA0 > TA3. Opération interdite (schéma `c_air_clim_changement`). Le producteur/propriétaire du schéma ne doit pas être un rôle de connexion.

Il devra préciser un propriétaire pour le schéma dans l'ordre de création qui soit un rôle de groupe dont il est membre :

```
CREATE SCHEMA c_air_clim_changement AUTHORIZATION g_snum ;
```

ASGARD retourne alors :

NOTICE: application des privilèges standards pour le rôle lecteur du schéma `c_air_clim_changement` :
 NOTICE: > GRANT USAGE ON SCHEMA `c_air_clim_changement` TO `g_consult`
 NOTICE: > GRANT SELECT ON ALL TABLES IN SCHEMA `c_air_clim_changement` TO `g_consult`
 NOTICE: > GRANT SELECT ON ALL SEQUENCES IN SCHEMA `c_air_clim_changement` TO `g_consult`
 NOTICE: ... Le schéma `c_air_clim_changement` apparaît désormais comme "créé" dans la table de gestion.
 CREATE SCHEMA

L'attribut `creation` du schéma est passé à `True`, les droits du rôle lecteur (ici `g_consult`, valeur par défaut renseignée lors de l'import de la nomenclature) sont appliqués et le producteur du schéma `c_air_clim_changement` (pré-renseigné à `g_admin` lors de l'import de la nomenclature) devient `g_snum`.

On peut constater que le groupe `g_consult` détient bien le privilège `USAGE` sur le schéma :

c_air_clim_changement			
General Sécurité Privileges par défaut SQL			
Droits			
	Bénéficiaire	Droits	Accordeur de droits
	g_consult	U	g_snum
	g_snum	UC	g_snum

Dans cet exemple, les commandes notifiées portant sur les objets n'ont pas d'effet, puisque le schéma est encore vide. Elles sont par contre utiles lorsque des objets sont créés simultanément au schéma, avec une commande telle que :

```
CREATE SCHEMA c_air_clim_changement AUTHORIZATION g_snum
CREATE TABLE temperature (id serial, jour date, valeur numeric) ;
```

III.3.A. CRÉATION D'UN SOUS-ENSEMBLE DE SCHÉMAS DE LA NOMENCLATURE

On peut bien sûr, schéma par schéma, passer l'attribut `creation` à `True` dans la vue `gestion_schema_usr`. Mais si les schémas considérés ont une caractéristique commune, telle que l'appartenance à un même domaine (`niv1`) ou un même sous-domaine (`niv2`), la création multiple pourra être réalisée avec une seule commande SQL.

EXEMPLE. Création de tous les schémas du domaine « eau » :

```
UPDATE z_asgard.gestion_schema_usr
SET creation = True
WHERE niv1 = 'eau' ;
```

ASGARD renvoi une succession de messages indiquant que les schémas ont bien été créés et les privilèges pré-définis appliqués.

```
NOTICE: ... Le schéma c_eau_aep a été créé.
NOTICE: application des privilèges standards pour le rôle lecteur du schéma c_eau_aep :
NOTICE: > GRANT USAGE ON SCHEMA c_eau_aep TO g_consult
NOTICE: > GRANT SELECT ON ALL TABLES IN SCHEMA c_eau_aep TO g_consult
NOTICE: > GRANT SELECT ON ALL SEQUENCES IN SCHEMA c_eau_aep TO g_consult
```

... et de même pour chaque schéma.

EXEMPLE. Création massive de tous les schémas de la nomenclature nationale :

```
UPDATE z_asgard.gestion_schema_usr
SET creation = True
WHERE nomenclature ;
```

III.4 — CRÉATION DE NOUVEAUX SCHÉMAS

ASGARD permet de créer un nouveau schéma en ajoutant un enregistrement dans la table de gestion (`INSERT`).

Les seuls champs obligatoires sont `nom_schema` et `producteur`.

Si creation n'est pas renseigné, il sera considéré comme valant False et le schéma sera référencé dans la table de gestion sans être créé. Pour procéder à sa création effective, il faudra ultérieurement basculer creation sur True.

Si creation vaut True lors de l'ajout de l'enregistrement, le schéma est immédiatement créé.

EXEMPLE. Création d'un schéma en passant par la vue gestion_schema_usr :

```
INSERT INTO z_asgard.gestion_schema_usr
  (nom_schema, creation, producteur) VALUES
  ('w_snow', true, 'g_snum') ;
```

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma w_snow (w).
NOTICE: ... Le schéma w_snow a été créé.
INSERT 0 1

Le producteur doit nécessairement être un rôle de groupe. Ainsi jon.snow (rôle de connexion) ne peut indiquer « jon.snow » dans le champ producteur.

Les autres champs peuvent également être précisés :

- ◆ bloc : s'il est renseigné, il servira de préfixe au nom du schéma.
- ◆ nomenclature : seul les membres de g_admin peuvent mettre ce champ à True.
- ◆ lecteur et editeur : si des rôles sont spécifiés pour ces champs, les droits correspondants leur seront attribués lors de la création du schéma.
- ◆ niv1, niv1_abr, niv2, niv2_abr : les champs niv1 et niv2 sont utilisés pour présenter un menu sous QGIS avec une hiérarchie niv1→niv2→ objets du schéma. Les champs niv1_abr et niv2_abr ne sont pas utilisés pour l'instant par ASGARD.

Toutes les méthodes classiques de création de schémas restent en outre pleinement utilisables dans le contexte d'ASGARD. Le nouveau schéma est alors enregistré dans la table de gestion avec comme producteur le rôle défini comme propriétaire par la commande de création.

EXEMPLE. Création par une commande CREATE :

```
CREATE SCHEMA w_snow AUTHORIZATION g_snum ;
```

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma w_snow (w).
NOTICE: ... Le schéma w_snow a été enregistré dans la table de gestion.
CREATE SCHEMA

ASGARD met alors à jour la table de gestion.

bloc character varying (1)	nom_schema character varying	creation boolean	producteur character varying	editeur character varying	lecteur character varying
w	w_snow	true	g_snum	[null]	[null]

ASGARD déduit le champ bloc (ici w) du nom du schéma, dès lors que celui-ci commence par une lettre – qui deviendra son bloc – suivie d'un tiret bas.

Une commande SQL `CREATE SCHEMA ... AUTHORIZATION [rôle]` échoue si le rôle spécifié n'existe pas. Par contre, si l'opérateur a renseigné dans la vue `gestion_schema_usr` un rôle producteur (mais aussi éditeur ou lecteur) non existant et que `creation` vaut `True`, alors ASGARD tente de créer le rôle de groupe manquant. Il est évidemment nécessaire que l'utilisateur courant soit habilité à créer des rôles (cf. I.3.C).

III.5 — SUPPRESSION DE SCHÉMAS

Pour des raisons de sécurité, la bascule directe du champ `creation` de `True` à `False` dans la vue `gestion_schema_usr` est interdite.

Cette bascule est par contre réalisée automatiquement par ASGARD lorsqu'une commande `DROP SCHEMA [CASCADE]` est lancée, de même que pour toute autre méthode de suppression extérieure à ASGARD.

EXEMPLE. Suppression du schéma `w_snow` et de tous les objets qu'il contient par une commande directe.

```
DROP SCHEMA w_snow CASCADE ;
```

NOTICE: ... La suppression du schéma `w_snow` a été enregistrée dans la table de gestion (`creation = False`).
`DROP SCHEMA`

Physiquement, le schéma est détruit (la commande s'exécute normalement), mais il reste enregistré avec `creation` valant `False` dans la table de gestion, pour permettre une hypothétique recréation. Ce système permet notamment de ne pas avoir à redéfinir les profils de droits en cas de sauvegarde/restauration d'un schéma.

Pour obtenir le même résultat en travaillant uniquement dans la table de gestion, il faudra passer d'abord l'attribut `bloc` du schéma à la valeur « d », qui représente la « corbeille » (cf. aussi III.2.A). Une fois le schéma dans la corbeille, il devient possible basculer `creation` sur `False`. ASGARD exécute alors un `DROP SCHEMA CASCADE` pour supprimer physiquement le schéma et tous les objets qu'il contient.

EXEMPLE. Suppression du schéma `w_snow` et de tous les objets qu'il contient via la table de gestion.

Premier temps, mise à la corbeille :

```
UPDATE z_asgard.gestion_schema_usr  
  SET bloc = 'd'  
  WHERE nom_schema = 'w_snow' ;
```

NOTICE: [table de gestion] Le schéma `w_snow` a été mis à la corbeille (bloc = 'd').

HINT: Si vous basculez `creation` sur `False`, le schéma et son contenu seront automatiquement supprimés.

UPDATE 1

Second temps, suppression par bascule de `creation` sur `False` :

```
UPDATE z_asgard.gestion_schema_usr  
  SET creation = False  
  WHERE nom_schema = 'w_snow' ;
```

NOTICE: ... La suppression du schéma `w_snow` a été enregistrée dans la table de gestion (`creation = False`).

NOTICE: ... Le schéma `w_snow` a été supprimé.

UPDATE 0

Cette dernière commande renverra toujours « UPDATE 0 », comme si elle n'avait eu aucun effet, même si un ou plusieurs enregistrements ont bien été modifiés en pratique (comme en attestent les messages). En effet, les mécanismes d'ASGARD font que l'UPDATE lancé par l'utilisateur est remplacé par une commande de suppression du schéma. C'est cette dernière qui entraînera la mise à jour effective du champ `creation`. En lui-même, le premier UPDATE n'affecte donc directement aucun enregistrement de la table de gestion.

Dans tous les cas la commande de suppression ne peut être lancée que par un membre de `g_admin` ou, plus largement, par un membre du rôle producteur du schéma qui soit doté du privilège `CREATE` sur la base et membre de l'éditeur de `z_asgard` (cf. III.1.B).

Si on le souhaite et si le schéma ne fait pas partie de la nomenclature nationale (`nomenclature = False`), l'enregistrement du schéma peut ensuite être retiré de la table de gestion.

```
DELETE FROM z_asgard.gestion_schema_usr WHERE nom_schema='w_snow';
```

DELETE 1

III.6 — MODIFICATIONS DE SCHÉMAS

III.6.A. RENOMMER UN SCHÉMA

Lorsque le nom d'un schéma est modifié via une commande `ALTER SCHEMA [...] RENAME TO`, ASGARD met à jour en conséquence la table de gestion avec le nouveau nom du schéma.

Réciproquement, un utilisateur peut renommer un schéma en changeant la valeur du champ `nom_schema` dans la vue `gestion_schema_usr`.

Ces deux méthodes sont parfaitement équivalentes. Dans les deux cas, l'utilisateur doit être habilité à renommer le schéma et à éditer la table de gestion²⁶.

EXEMPLE. Modification du nom du schéma `w_snow`, renommé en `w_jon_snow`.

Avec une commande classique :

```
ALTER SCHEMA w_snow RENAME TO w_jon_snow ;
```

NOTICE: ... Le nom du schéma `w_jon_snow` a été mis à jour dans la table de gestion.
`ALTER SCHEMA`

Par la table de gestion :

```
UPDATE z_asgard_admin.gestion_schema
  SET nom_schema = 'w_jon_snow'
  WHERE nom_schema = 'w_snow' ;
```

NOTICE: ... Le schéma `w_jon_snow` a été renommé.
`UPDATE 1`

III.6.B. MODIFIER LE PROPRIÉTAIRE D'UN SCHÉMA

La modification du propriétaire du schéma par une commande `ALTER SCHEMA [...] OWNER TO` est prise en charge de la façon suivante :

- ◆ Le nouveau propriétaire est renseigné comme producteur dans la table de gestion.

²⁶ Membre de `g_admin`, ou à la fois membre du rôle producteur du schéma, disposant du privilège `CREATE` sur la base et membre de l'éditeur de `z_asgard` (cf. III.1.B).

- ◆ Il est fait propriétaire de tous les objets contenus dans le schéma.

L'utilisateur est informé de toutes ces actions par une série de notifications.

Lorsque le producteur est modifié dans `gestion_schema_usr`, le rôle désigné devient automatiquement propriétaire du schéma et de tous les objets qu'il contient.

Là encore, les deux méthodes mènent exactement au même résultat et l'utilisateur doit disposer des mêmes privilèges dans les deux cas²⁷.

EXEMPLE. Modification du propriétaire du schéma `w_snow`, qui contient pour seuls objets une table et la séquence utilisée par sa clé primaire.

Avec une commande classique :

```
ALTER SCHEMA w_snow OWNER TO g_admin ;
```

```
NOTICE: attribution de la propriété des objets au rôle producteur du schéma w_snow :
NOTICE: > ALTER TABLE w_snow.journal_du_mur OWNER TO g_admin
NOTICE: > ALTER SEQUENCE w_snow.journal_du_mur_id_seq OWNER TO g_admin
NOTICE: ... Le propriétaire du schéma w_snow a été mis à jour dans la table de gestion.
ALTER SCHEMA
```

Par l'intermédiaire de la table de gestion :

```
UPDATE z_asgard.gestion_schema_usr
SET producteur = 'g_admin'
WHERE nom_schema = 'w_snow' ;
```

```
NOTICE: attribution de la propriété du schéma et des objets au rôle producteur du schéma w_snow :
NOTICE: > ALTER SCHEMA w_snow OWNER TO g_admin
NOTICE: > ALTER TABLE w_snow.journal_du_mur OWNER TO g_admin
NOTICE: > ALTER SEQUENCE w_snow.journal_du_mur_id_seq OWNER TO g_admin
UPDATE 1
```

III.6.C. MODIFIER LES PRIVILÈGES

Indépendamment d'ASGARD, lorsque le propriétaire d'un objet change, PostgreSQL attribue au nouveau propriétaire très exactement les mêmes privilèges sur cet objet que ceux qu'avait l'ancien, et retire à ce dernier tous ses privilèges. Ainsi, le nouveau propriétaire n'a pas nécessairement tous les droits sur les objets qu'il possède. En tant que propriétaire, il est évidemment habilité à rétablir ses privilèges manquants s'il le souhaite, de la même façon qu'il peut en allouer ou en retirer aux autres rôles.

²⁷ Membre de `g_admin`, ou à la fois membre de l'ancien rôle producteur du schéma, membre du nouveau producteur, disposant du privilège `CREATE` sur la base et membre de l'éditeur de `z_asgard` (cf. III.1.B).

ASGARD s'inspire de ce comportement pour les changements de lecteur et d'éditeur. Sous réserve que l'opération n'ait pas pour objet de supprimer l'éditeur/lecteur ou d'en déclarer un alors qu'il n'y en avait pas précédemment, les privilèges attribués au nouveau rôle sont très exactement ceux qu'avait l'ancien, y compris s'ils diffèrent des privilèges standards pour la fonction.

Lorsqu'un nouvel éditeur ou lecteur est désigné, les mécanismes automatiques d'ASGARD lui transfèrent à l'identique les privilèges de l'ancien, et PostgreSQL fait de même pour les propriétaires.

D'une manière générale, les mécanismes automatiques d'ASGARD respectent les modifications manuelles sur les privilèges et ils ne s'intéressent qu'aux droits des rôles producteurs, éditeurs et lecteurs.

Si l'utilisateur ne souhaite pas conserver certaines modifications manuelles, il lui est possible :

- ◆ de les reprendre à la main, par des commandes GRANT/REVOKE ;
- ◆ d'utiliser la fonction `asgard_initialise_schema` pour rétablir les privilèges standards sur le schéma et tout son contenu ;
- ◆ d'utiliser la fonction `asgard_initialise_obj` pour rétablir les privilèges standards sur un objet donné.

Les lecteurs et éditeurs ne peuvent être modifiés que par saisie dans la table de gestion.

EXEMPLE. Désignation du pseudo-rôle public comme lecteur du schéma `w_snow` en remplacement de `gb_consult` :

```
UPDATE z_asgard.gestion_schema_usr
SET lecteur = 'public'
WHERE nom_schema = 'w_snow' ;
```

```
NOTICE: suppression et transfert vers le nouveau lecteur des privilèges de l'ancien lecteur du schéma
w_snow :
NOTICE: > GRANT USAGE ON SCHEMA w_snow TO public
NOTICE: > REVOKE USAGE ON SCHEMA w_snow FROM gb_consult
NOTICE: > GRANT SELECT ON TABLE w_snow.journal_du_mur TO public
NOTICE: > REVOKE SELECT ON TABLE w_snow.journal_du_mur FROM gb_consult
NOTICE: > GRANT SELECT ON SEQUENCE w_snow.journal_du_mur_id_seq TO public
NOTICE: > REVOKE SELECT ON SEQUENCE w_snow.journal_du_mur_id_seq FROM gb_consult
UPDATE 1
```

III.7 — CRÉATION D'UN OBJET DANS UN SCHÉMA

La création d'objets dans les schémas se fait selon les possibilités habituelles (création directe en SQL, import/export, COPY, etc.). ASGARD se contente d'intercepter les créations pour réaffecter la propriété des objets au rôle de groupe producteur du schéma et, lorsqu'il y a lieu, attribuer les droits standards de l'éditeur et du lecteur selon les principes décrits au II.1 (si le schéma est référencé par ASGARD).

EXEMPLE. Lorsque jon.snow, membre de g_snum, crée la table journal_du_mur dans le schéma w_snow dont g_snum est le producteur et public le lecteur, g_snum devient automatiquement propriétaire de la table et les droits de lecture usuels sont conférés à public :

```
CREATE TABLE w_snow.journal_du_mur (id serial PRIMARY KEY, jour date,
entree text) ;
```

```
NOTICE: réattribution de la propriété de w_snow.journal_du_mur au rôle producteur du schéma :
NOTICE: > ALTER table w_snow.journal_du_mur OWNER TO g_snum
NOTICE: application des privilèges standards pour le rôle lecteur du schéma :
NOTICE: > GRANT SELECT ON TABLE w_snow.journal_du_mur TO public
NOTICE: application des privilèges standards pour le rôle lecteur du schéma :
NOTICE: > GRANT SELECT ON SEQUENCE w_snow.journal_du_mur_id_seq TO public
CREATE TABLE
```

III.8 — MODIFICATION DES OBJETS

ASGARD référence les schémas, mais pas les objets qu'ils contiennent, ce qui le rend relativement peu efficace quand il s'agit de suivre leurs changements d'état.

Il interagit uniquement avec les commandes ALTER :

- ◆ qui visent des objets rattachés à un schéma référencé dans la table de gestion ;
- ◆ à l'issue desquelles le propriétaire de l'objet n'est plus le rôle producteur du schéma qui le contient.

Et sa seule action est d'attribuer ou réattribuer la propriété de l'objet en question au producteur du schéma.

EXEMPLE. Lorsque la table journal_du_mur est déplacée de w_snow (producteur g_snum) à c_bibliotheque (producteur g_admin), g_admin obtient la propriété de la table :

```
ALTER TABLE w_snow.journal_du_mur SET SCHEMA c_bibliotheque ;
```

NOTICE: attribution de la propriété de c_bibliotheque.journal_du_mur au rôle producteur du schéma :
NOTICE: > ALTER table c_bibliotheque.journal_du_mur OWNER TO g_admin
ALTER TABLE

Dans ces conditions, les commandes ALTER [OBJET] ... OWNER TO qui viseraient à définir un propriétaire différent du producteur du schéma sont inhibées : après exécution de la commande, ASGARD rend la propriété de l'objet au producteur.

EXEMPLE. Tentative pour attribuer à jon.snow la propriété de la table journal_du_mur.

```
ALTER TABLE w_snow.journal_du_mur OWNER TO "jon.snow" ;
```

NOTICE: attribution de la propriété de w_snow.journal_du_mur au rôle producteur du schéma :
NOTICE: > ALTER table w_snow.journal_du_mur OWNER TO g_snum
ALTER TABLE

ASGARD peut repérer qu'un objet n'a plus le bon propriétaire, mais il lui est impossible de distinguer spécifiquement les commandes ALTER [OBJET] ... SET SCHEMA parmi l'ensemble des commandes ALTER [OBJET]. Dès lors, il n'est pas en mesure de transférer automatiquement les privilèges des éditeur/lecteur de l'ancien schéma à ceux du nouveau. Très concrètement, hormis pour l'ancien et le nouveau producteur (entre lesquels les privilèges sont transférés), les droits sur un objet restent inchangés lorsqu'il est déplacé par une commande ALTER.

Lorsqu'un objet change de schéma, les privilèges de l'éditeur et du lecteur ne sont pas modifiés en conséquence, sauf à déplacer l'objet avec la fonction `asgard_deplace_obj`.

Pour une gestion plus élaborée des droits lors d'un changement de schéma, il est recommandé de procéder au déplacement en utilisant la fonction `asgard_deplace_obj`.

Il est également possible de déplacer l'objet avec une commande SQL standard puis de recourir a posteriori à la fonction `asgard_initialise_obj` afin de nettoyer les droits sur l'objet déplacé et revenir aux privilèges standards²⁸.

28 Si l'objet déplacé est une table avec un champ serial, il faudra alors penser à appliquer également la fonction `asgard_initialise_obj` sur la séquence associée – ou utiliser `asgard_initialise_schema` pour réinitialiser les droits sur tous les objets du schéma.

On remarquera que la question ne se pose que pour les objets sur lesquels il est possible de conférer des privilèges. Pour les conversions, collationnements, opérateurs, dictionnaires et configuration de recherche plein texte, la question ne se pose pas, il convient et suffit d'utiliser une commande `ALTER [OBJET] ... SET SCHEMA` pour changer l'objet de schéma. ASGARD prendra soin d'attribuer la propriété de l'objet au producteur du schéma de destination.

III.9 — OBJETS NON PRIS EN CHARGE

À ce stade, les types d'objets suivants ne sont pas gérés par ASGARD :

- ◆ objet statistique étendu (extended statistics) ;
- ◆ classe d'opérateurs (operator class) ;
- ◆ famille d'opérateurs (operator family).

La cohérence de leur propriétaire avec celui du schéma qui les contient n'est pas contrôlée en cas de modification du propriétaire de l'objet et/ou de producteur du schéma. Ils sont toutefois considérés par la fonction de diagnostic (IV.12). L'attribution de privilèges n'étant de toute façon pas prévue par PostgreSQL sur ces objets, les limitations d'ASGARD sont sans conséquence de ce point de vue.

Tous les autres objets supportés par les versions 9.5 à 10 de PostgreSQL, qui sont rattachés à un schéma (au contraire, par exemple, des serveurs distants) et ont un propriétaire paramétrable (au contraire, par exemple, des analyseurs et modèles de recherche plein texte) sont pris en charge.

III.10 — MODIFICATION DES RÔLES

Au contraire des noms des schémas, les noms des rôles désignés comme producteur, éditeur ou lecteur ne sont pas mis à jour automatiquement dans la table de gestion lorsque ces rôles sont renommés (`ALTER ROLE [...] RENAME TO`) ou supprimés²⁹.

Cependant, lorsqu'un utilisateur intervient sur une ligne de la table de gestion se rapportant à un schéma créé et comportant un nom de rôle obsolète ou inexistant, le champ est actualisé.

29 Étant entendu que la suppression ne concerne que les éditeurs et lecteur (après révocation de tous leurs privilèges), PostgreSQL ne permettant pas de supprimer un rôle qui serait propriétaire d'un schéma.

EXEMPLE. Le producteur du schéma w_snow est renommé :

```
ALTER ROLE g_snum RENAME TO g_sg_snum ;
```

Sur le moment, rien ne se passe dans la table de gestion.

nom_schema character varying	producteur character varying	editeur character varying	lecteur character varying
w_snow	g_snum	[null]	gb_consult

Toutefois, lorsqu'un utilisateur modifie – par exemple – le rôle désigné comme lecteur pour le schéma, il est informé de la mise à jour du nom du producteur (deux premiers messages de la série).

```
UPDATE z_asgard.gestion_schema_usr
SET lecteur = 'gb_consult'
WHERE nom_schema = 'w_snow' ;
```

NOTICE: [table de gestion] Schéma w_snow. Mise à jour du libellé du rôle producteur, renommé entre temps

DETAIL: Ancien nom "g_snum", nouveau nom "g_sg_snum".

NOTICE: suppression et transfert vers le nouveau lecteur des privilèges de l'ancien lecteur du schéma w_snow :

NOTICE: > GRANT USAGE ON SCHEMA w_snow TO gb_consult

NOTICE: > REVOKE USAGE ON SCHEMA w_snow FROM public

UPDATE 1

nom_schema character varying	producteur character varying	editeur character varying	lecteur character varying
w_snow	g_sg_snum	[null]	gb_consult

Lorsqu'un rôle lecteur ou éditeur n'existe plus, le champ est vidé.

Pour mettre à jour en une seule fois toute la table de gestion après la modification d'un rôle, l'ADL pourra utiliser la fonction asgard_nettoyage_roles.

Lorsque le but poursuivi est la suppression du rôle, il est conseillé d'utiliser la fonction asgard_reaffecte_role, qui propose une méthode rapide et compatible avec ASGARD pour supprimer l'ensemble des droits d'un rôle sur les objets de la base – ou les réaffecter à un autre rôle. Dans ce cas, aucune intervention a posteriori sur la table de gestion n'est nécessaire.

Tout ceci ne concerne cependant que les schémas créés, pour lesquels ASGARD identifie le rôle non par son nom mais par l'identifiant système (OID) invariant que lui attribue PostgreSQL.

Pour les schémas non créés, dont les rôles peuvent ne pas encore exister, ASGARD n'a aucun moyen de distinguer un renommage d'un changement de rôle et ne procédera donc, quoi qu'il arrive, à aucune modification. Il est du ressort de l'ADL de s'assurer que les noms pré-renseignés restent conformes à ses attentes et de réaliser les corrections nécessaires par de simples UPDATE sur `gestion_schema_usr`.

III.11 — SAUVEGARDE ET RESTAURATION DE LA BASE

Lorsque la base de données est sauvegardée, le fichier de sauvegarde contient :

- ♦ une commande `CREATE EXTENSION asgard`, qui lors de la restauration, réinstallera l'extension ;
- ♦ les données des tables `gestion_schema` et `asgard_parametre`.

Si la structure de certains objets de l'extension a été modifiée, ils seront recréés dans leur état d'origine, tel que défini par le script de l'extension, et les modifications seront perdues.

Les informations saisies dans la table de gestion et la table de paramétrage sont par contre intégralement conservées.

À noter que les sauvegardes de schémas via `pg_dump` ne fonctionneront pas sur les schémas d'ASGARD `z_asgard` et `z_asgard_admin` – les fichiers obtenus ne contiendront aucune information.

III.12 — NETTOYAGE DES DROITS ET RÉOLUTION DES PROBLÈMES

ASGARD met à disposition des ADL une petite gamme de fonctions pour identifier et résoudre leurs problèmes de droits plus rapidement qu'avec les utilitaires de pgAdmin ou les commandes SQL classiques.

`asgard_diagnostic` (cf. IV.12) repère tous les écarts vis-à-vis des droits standards prévus par ASGARD. Appliquer cette fonction est le premier réflexe à avoir lorsque des utilisateurs n'accèdent pas à des objets qu'ils devraient pouvoir utiliser (ou inversement).

En une seule commande, `asgard_initialise_all_schemas` (cf. IV.11) permet de revenir aux droits standards d'ASGARD pour l'ensemble des schémas référencés par ASGARD : les producteurs, éditeurs et lecteurs retrouvent leurs droits normaux, ceux des autres rôles sont révoqués. Pour un service qui n'aurait pas personnalisé les droits au-delà de la désignation des producteurs, éditeurs et lecteurs, c'est la meilleure manière de nettoyer sa base.

`asgard_initialise_schema` (cf. IV.3) et `asgard_initialise_obj` (cf. IV.4) suivent le même principe, mais respectivement à l'échelle d'un schéma et d'un objet contenu dans un schéma. Elles permettent de procéder à des corrections plus localisées.

IV — FONCTIONS UTILITAIRES

IV.1 — PRÉSENTATION

Des fonctions utilitaires sont mises à dispositions dans le schéma `z_asgard` ou, pour trois d'entre elles à l'usage plus délicat, `z_asgard_admin`.

Pour l'heure, elles doivent être appelées par des commandes SQL dans un éditeur de requêtes (pgAdmin ou DBManager), généralement sous la forme :

```
SELECT fonction( [ argument [, ...] ] ) ;
```

où

fonction est le nom de la fonction

argument est la liste de ses arguments éventuels

Quelques fonctions, qui renvoient une table à plusieurs attributs, gagneront à être lancées comme suit pour un résultat plus lisible :

```
SELECT { * | expression [, ...] }  
FROM fonction( [ argument [, ...] ] ) ;
```

où

fonction est le nom de la fonction

argument est la liste de ses arguments éventuels

expression est un appel aux attributs du résultat de la fonction, sinon * pour tous les attributs

Outre les fonctions décrites dans la présente partie, le schéma `z_asgard` contient des fonctions qui ont été conçues pour les mécanismes internes d'ASGARD. Elles n'ont a priori pas d'intérêt immédiat pour la gestion courante de la base, mais il n'est pas interdit de s'en servir et elle pourraient être utiles dans certains cas spécifiques. Ces fonctions supplémentaires sont présentées en annexe A.



fonctions d'ASGARD

◆ z_asgard_admin

réservé à l'ADL (membres de g_admin)

☞ fonctions utilitaires à l'usage de l'ADL

☞ asgard_all_login_grant_role

donne permission sur un rôle à tous les rôles de connexion du serveur

☞ asgard_import_nomenclature

pré-référence les schémas de la nomenclature thématique nationale dans la table de gestion

☞ asgard_initialisation_gestion_schema

référence les schémas pré-existants de la base dans la table de gestion

☞ asgard_reaffecte_role

supprime tous les droits d'un rôle sur les objets de la base ou les réassigne à un autre rôle

☞ asgard_sortie_gestion_schema

déréférence un schéma de la table de gestion

☞ asgard_initialise_all_schemas

réinitialise les droits sur tous les schémas référencés (efface les modifications manuelles)

☞ asgard_diagnostic

examine tous les schémas référencés et liste les écarts aux droits standards prévus par ASGARD

☞ fonctions déclencheurs

- implémentent les mécanismes d'ASGARD : contrôle des informations saisies dans la table de gestion, interception des commandes, application automatique des droits...

◆ z_asgard

accessible à tous les utilisateurs (membres de g_consult)

☞ fonctions utilitaires à l'usage de l'ADL, des administrateurs délégués et des producteurs des schémas

☞ asgard_deplace_obj

change un objet de schéma avec une gestion des droits adaptée

☞ asgard_initialise_obj

réinitialise les droits sur un objet (efface les modifications manuelles)

☞ asgard_initialise_schema

réinitialise les droits sur un schéma et son contenu (efface les modifications manuelles)

☞ asgard_nettoyage_roles

met à jour la table de gestion après le renommage ou la suppression d'un rôle

☞ autres fonctions

cf. annexe A

- appelées par les fonctions déclencheurs d'ASGARD ou les fonctions utilitaires
- ne sont pas pensées pour un usage manuel, mais ce n'est pas interdit

IV.2 — ASGARD_DEPLACE_OBJ

```
z_asgard.asgard_deplace_obj(obj_schema text, obj_nom text, obj_typ text,  
schema_cible text [, variante integer DEFAULT 1 ] )
```

La fonction `asgard_deplace_obj` permet de déplacer un objet vers un nouveau schéma en spécifiant la gestion voulue sur les droits de l'objet : transfert ou réinitialisation plus ou moins complète des privilèges.

C'est un outil essentiel pour les tables et assimilées, ou encore les séquences, pour lesquelles un déplacement par la commande `ALTER [OBJET] ... SET SCHEMA` ne produira presque jamais le résultat escompté sur les droits (cf. III.8). Pour les fonctions, agrégats, types et domaines, sur lesquels ASGARD n'accorde pas de privilège spécifique aux lecteurs et éditeurs, la plus-value est beaucoup plus réduite et, sauf à vouloir déplacer l'objet et réinitialiser les droits en une seule commande (avec la variante 2, cf. ci-après), une commande SQL classique sera souvent tout aussi efficace.

Si l'objet est une table avec un ou plusieurs champs `serial` ou `IDENTITY`³⁰, les privilèges sur les séquences associées³¹ seront traités selon les mêmes modalités que ceux de la table.

Les autres objets qui seraient liés à l'objet déplacé ne sont pas considérés pour l'heure : les privilèges de l'ancien propriétaire seront transférés sur le nouveau et tous les autres privilèges resteront inchangés.

SCHÉMA : `z_asgard`.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard.asgard_deplace_obj(obj_schema, obj_nom, obj_typ,  
schema_cible [, variante ] ) ;
```

Les arguments doivent être spécifiés dans l'ordre.

Les arguments optionnels peuvent être omis, mais seulement s'ils sont à la fin de la liste.

³⁰ À partir de la version 10 de PostgreSQL.

³¹ Dont PostgreSQL change le schéma et le propriétaire selon ceux de la table dont elles dépendent.

Avec des arguments nommés :

```
SELECT z_asgard.asgard_deplace_obj(  
    obj_schema := obj_schema,  
    obj_nom := obj_nom,  
    obj_typ := obj_typ,  
    schema_cible := schema_cible [,  
    variante := variante ]  
);
```

L'ordre des arguments nommés n'a pas d'importance.
Les arguments optionnels peuvent être omis.

Il est bien entendu possible de commencer avec des arguments positionnels (dans l'ordre) et terminer par des arguments nommés.

ARGUMENTS :

- (1) **obj_schema** est une chaîne de caractères correspondant au nom du schéma contenant l'objet à déplacer ;
- (2) **obj_nom** est une chaîne de caractères correspondant au nom de l'objet ;
- (3) **obj_typ** est une chaîne de caractères correspondant au type de l'objet. Valeurs acceptées : « table », « partitioned table » (sera assimilé à « table » et peut être écrit comme tel), « view », « materialized view », « foreign table », « sequence », « function », « aggregate », « type », « domain » ;
- (4) **schema_cible** est une chaîne de caractères correspondant au nom du schéma où doit être déplacé l'objet ;
- (5) [optionnel] **variante** est un nombre entier compris entre 1 et 6, qui définit le comportement attendu par l'utilisateur vis-à-vis des privilèges. Valeur par défaut 1.

Les valeurs de **obj_schema** et **schema_cible** s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'c_Bibliothèque' et non '"c_Bibliothèque"'.

C'est également vrai pour **obj_nom**, sauf dans le cas des **obj_typ** fonction et aggregate. Pour les **obj_typ** fonction et aggregate, les guillemets devront impérativement être présents dans **obj_nom** si le nom n'est pas normalisé – et seulement dans ce cas. Ainsi, on écrira 'suite_arithmetique(integer,integer)' (normalisé, sans guillemets), mais '"Suite Arithmétique"(integer,integer)' (non normalisé, avec guillemets).

Lorsque l'objet à déplacer est une fonction ou un agrégat, **obj_nom** doit spécifier les types de ses arguments, sous forme non abrégée (integer et pas int) et, s'il y en a plusieurs, sans espace après les virgules.

VARIANTES :

Le tableau suivant détaille l'effet de la fonction sur les droits appliqués à l'objet selon la valeur de l'argument *variante*.

	Droits du producteur	Droits des lecteur et éditeur (si définis)	Droits des autres rôles (le cas échéant)
1	Transférés de l'ancien producteur au nouveau.	Transférés si un éditeur (ou lecteur) était désigné pour l'ancien schéma, sinon application des privilèges standard pour la fonction.	Conservés à l'identique.
2	Réinitialisés ³² selon les privilèges standards.	Réinitialisés selon les privilèges standards.	Supprimés.
3	Transférés de l'ancien producteur au nouveau.	Transférés si un éditeur (ou lecteur) était désigné pour l'ancien schéma, sinon application des privilèges standard pour la fonction.	Supprimés.
4	Transférés de l'ancien producteur au nouveau.	Conservés à l'identique.	Conservés à l'identique.
5	Transférés de l'ancien producteur au nouveau.	Réinitialisés selon les privilèges standards.	Supprimés.
6	Réinitialisés ³³ selon les privilèges standards.	Réinitialisés selon les privilèges standards.	Conservés à l'identique.

On pourra noter que la variante 4 équivaut à une commande ALTER [OBJET] ... SET SCHEMA. Toutes les autres sont absolument équivalentes entre elles dès lors qu'aucune modification manuelle n'a été réalisée.

La variante 1 (valeur par défaut) adopte le comportement canonique d'ASGARD : transfert des privilèges pour les rôles producteur, éditeur et lecteur, et conservation pour les autres rôles.

La variante 2 est une remise à zéro complète des privilèges.

32 Les privilèges sont d'abord transférés (par PostgreSQL, lors de l'exécution de la commande de réassignation du propriétaire), puis explicitement réinitialisés par la fonction dans un second temps.

33 Idem.

RÉSULTAT : « __ DEPLACEMENT REUSSI. » si la requête s’est exécutée normalement. Le détail des commandes GRANT et REVOKE exécutées apparaît dans l’onglet « Messages » de l’éditeur de requêtes de pgAdmin.

EXEMPLE. Avec la variante n°2, qui revient aux droits standards :

La table `journal_du_mur`, dont la clé primaire est un champ `serial`, est déplacée de `w_snow` (producteur `g_snum`, pas d’éditeur, lecteur `g_consult`) à `c_bibliotheque` (producteur `g_admin`, éditeur `g_snum`, lecteur `public`). Le privilège `DELETE` avait été manuellement révoqué pour le producteur de l’ancien schéma.

```
SELECT z_asgard.asgard_deplace_obj('w_snow', 'journal_du_mur', 'table',
'c_bibliotheque', 2) ;
```

Données	EXPLAIN	Messages	Notifications
asgard_deplace_obj text			
1		__ DEPLACEMENT REUSSI.	

```
NOTICE: attribution de la propriété de c_bibliotheque.journal_du_mur au rôle producteur du schéma :
NOTICE: > ALTER table c_bibliotheque.journal_du_mur OWNER TO g_admin
NOTICE: ... Objet déplacé dans le schéma c_bibliotheque.
NOTICE: application des privilèges standards pour le rôle éditeur du schéma :
NOTICE: > GRANT SELECT, UPDATE, DELETE, INSERT ON TABLE c_bibliotheque.journal_du_mur TO
g_snum
NOTICE: > GRANT SELECT, USAGE ON SEQUENCE c_bibliotheque.journal_du_mur_id_seq TO g_snum
NOTICE: suppression des privilèges de l'ancien lecteur, g_consult :
NOTICE: > REVOKE SELECT ON TABLE c_bibliotheque.journal_du_mur FROM g_consult
NOTICE: > REVOKE SELECT ON SEQUENCE c_bibliotheque.journal_du_mur_id_seq FROM g_consult
NOTICE: application des privilèges standards pour le rôle lecteur du schéma :
NOTICE: > GRANT SELECT ON TABLE c_bibliotheque.journal_du_mur TO public
NOTICE: > GRANT SELECT ON SEQUENCE c_bibliotheque.journal_du_mur_id_seq TO public
```

IV.3 — ASGARD_INITIALISE_SCHEMA

```
z_asgard.asgard_initialise_schema(n_schema text [, b_preserve boolean
DEFAULT false [, b_gs boolean DEFAULT false ]]
```

La fonction `asgard_initialise_schema` permet de réinitialiser les droits sur un schéma. Pour les rôles producteur, éditeur et lecteur désignés dans la table de gestion, elle revient aux privilèges standards. Les privilèges des autres rôles sont effacés³⁴, ainsi que les

³⁴ Hors privilèges implicites du pseudo-rôle `public` (`USAGE` sur les fonctions et les types).

privilèges par défaut définis sur le schéma considéré par la commande ALTER DEFAULT PRIVILEGES.

Si elle est appliquée à un schéma existant non référencé dans la table de gestion, elle l'ajoute (cf. paramètre n°3 pour le choix du producteur et n°2 pour la conservation ou non des privilèges pré-définis). Le producteur du schéma deviendra également producteur de tous les objets qu'il contient, s'il ne l'était pas déjà.

La fonction échoue si le schéma donné en argument n'existe pas.

Lorsqu'elle est appliquée aux schémas z_asgard et z_asgard_admin, asgard_initialise_schema préserve les privilèges de g_admin_ext et g_consult nécessaires au bon fonctionnement d'ASGARD³⁵. Elle peut également être utilisée pour les rétablir en cas de suppression accidentelle, mais la fonction asgard_initialise_all_schemas (variante n°2) sera plus efficace pour les situations vraiment critiques.

SCHÉMA : z_asgard.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard.asgard_initialise_schema(n_schema [, b_preserve [,  
b_gs ]]) ;
```

Les arguments doivent être spécifiés dans l'ordre.
Les arguments optionnels peuvent être omis, mais seulement s'ils sont à la fin de la liste.

Avec des arguments nommés :

```
SELECT z_asgard.asgard_initialise_schema(  
    n_schema := n_schema [,  
    b_preserve := b_preserve [,  
    b_gs := b_gs ]]  
) ;
```

L'ordre des arguments nommés n'a pas d'importance.
Les arguments optionnels peuvent être omis.

Il est bien entendu possible de commencer avec des arguments positionnels (dans l'ordre) et terminer par des arguments nommés.

ARGUMENTS :

³⁵ Très précisément, elle les supprime puis les restaure.

- (1) ***n_schema*** est une chaîne de caractères correspondant au nom d'un schéma présumé existant. Les valeurs de ***n_schema*** s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'c_Bibliothèque' et non '"c_Bibliothèque"' ;
- (2) [optionnel] ***b_preserve*** est un paramètre booléen. Pour un schéma encore non référencé (ou pré-référencé comme non-créé) dans la table de gestion, une valeur True signifie que les privilèges des rôles lecteur et éditeur doivent être ajoutés par-dessus les droits actuels. Avec la valeur par défaut False, les privilèges sont réinitialisés. Ce paramètre est ignoré pour un schéma déjà référencé (et les privilèges sont réinitialisés) ;
- (3) [optionnel] ***b_gs*** est un booléen indiquant si, dans l'hypothèse où un schéma serait déjà référencé – nécessairement comme non créé – dans la table de gestion, c'est le propriétaire du schéma qui doit devenir le producteur (False) ou le producteur pré-renseigné dans la table de gestion qui doit devenir le propriétaire du schéma (True). False par défaut. Ce paramètre est ignoré pour un schéma déjà référencé.

RÉSULTAT : « __ REINITIALISATION REUSSIE. », ou « __ INITIALISATION REUSSIE. » pour un schéma non référencé comme créé et pour lequel le second argument vaut True.

EXEMPLE. Utilisation d'asgard_initialise_schema pour référencer un schéma pré-existant dans la table de gestion, en l'occurrence c_bibliotheque.

Si on lance la fonction sans argument optionnel...

```
SELECT z_asgard.asgard_initialise_schema('c_bibliotheque') ;
```

Données	EXPLAIN	Messages	Notifications
asgard_initialise_schema text			
1		__ REINITIALISATION REUSSIE.	

```
NOTICE: [table de gestion] Mise à jour du bloc pour le schéma c_bibliotheque (c).
NOTICE: (schéma c_bibliotheque pré-existant)
NOTICE: attribution de la propriété des objets au rôle producteur du schéma c_bibliotheque :
NOTICE: > ALTER TABLE c_bibliotheque.journal_du_mur OWNER TO g_admin
NOTICE: > ALTER SEQUENCE c_bibliotheque.journal_du_mur_id_seq OWNER TO g_admin
NOTICE: ... Le schéma c_bibliotheque a été enregistré dans la table de gestion.
NOTICE: remise à zéro des privilèges manuels du pseudo-rôle public :
NOTICE: > néant
NOTICE: remise à zéro des privilèges des autres rôles (pour le producteur, les éventuels privilèges
manquants sont réattribués) :
NOTICE: > REVOKE USAGE ON SCHEMA c_bibliotheque FROM "jon.snow"
NOTICE: > REVOKE SELECT ON TABLE c_bibliotheque.journal_du_mur FROM "jon.snow"
NOTICE: > REVOKE INSERT ON TABLE c_bibliotheque.journal_du_mur FROM "jon.snow"
```

... on constate que les privilèges de `jon.snow` ont été révoqués, ce qui n'est peut-être pas souhaitable (cf. II.3.A). Dans ce cas, il aurait été préférable de spécifier dès le départ que les privilèges pré-existants devaient être conservés :

```
SELECT z_asgard.asgard_initialise_schema('c_bibliotheque', True) ;
```

Données	EXPLAIN	Messages	Notifications
<div>asgard_initialise_schema</div> <div>text</div>			
1		__ INITIALISATION REUSSIE.	

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma c_bibliotheque (c).

NOTICE: (schéma c_bibliotheque pré-existant)

NOTICE: attribution de la propriété des objets au rôle producteur du schéma c_bibliotheque :

NOTICE: > ALTER TABLE c_bibliotheque.journal_du_mur OWNER TO g_admin

NOTICE: > ALTER SEQUENCE c_bibliotheque.journal_du_mur_id_seq OWNER TO g_admin

NOTICE: ... Le schéma c_bibliotheque a été enregistré dans la table de gestion.

EXEMPLE. Utilisation d'`asgard_initialise_schema` pour effacer des modifications manuelles de privilèges sur un schéma déjà référencé. Dans cet exemple, on considère le schéma `w_snow`, sur lequel des droits avaient été accordés au rôle de connexion `jon.snow` sans qu'il soit identifié ni comme lecteur, ni comme éditeur (et il n'y a d'ailleurs ni lecteur ni éditeur désigné pour ce schéma).

```
SELECT z_asgard.asgard_initialise_schema('w_snow') ;
```

Données	EXPLAIN	Messages	Notifications
<div>asgard_initialise_schema</div> <div>text</div>			
1		__ REINITIALISATION REUSSIE.	

NOTICE: (ré)attribution de la propriété des objets au rôle producteur du schéma :

NOTICE: > néant

NOTICE: remise à zéro des privilèges manuels du pseudo-rôle public :

NOTICE: > néant

NOTICE: remise à zéro des privilèges des autres rôles (pour le producteur, les éventuels privilèges manquants sont réattribués) :

NOTICE: > REVOKE USAGE ON SCHEMA w_snow FROM "jon.snow"

NOTICE: > REVOKE SELECT ON TABLE w_snow.journal_du_mur FROM "jon.snow"

NOTICE: > REVOKE INSERT ON TABLE w_snow.journal_du_mur FROM "jon.snow"

IV.4 — ASGARD_INITIALISE_OBJ

```
z_asgard.asgard_initialise_obj(obj_schema text, obj_nom text, obj_typ
text)
```

La fonction `asgard_initialise_obj` permet de réinitialiser les droits sur un objet. Pour les rôles producteur, éditeur et lecteur désignés dans la table de gestion pour son schéma, elle revient aux privilèges standards. Les privilèges des autres rôles sur l'objet sont effacés³⁶.

SCHÉMA : `z_asgard`.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard.asgard_initialise_obj(obj_schema, obj_nom, obj_typ) ;
```

Les arguments doivent être spécifiés dans l'ordre.

Avec des arguments nommés :

```
SELECT z_asgard.asgard_initialise_obj(
    obj_schema := obj_schema,
    obj_nom := obj_nom,
    obj_typ := obj_typ
) ;
```

L'ordre des arguments nommés n'a pas d'importance.

Il est bien entendu possible de commencer avec des arguments positionnels (dans l'ordre) et terminer par des arguments nommés.

ARGUMENTS :

- (1) ***obj_schema*** est une chaîne de caractères correspondant au nom du schéma contenant l'objet ;
- (2) ***obj_nom*** est une chaîne de caractères correspondant au nom de l'objet ;

³⁶ Hors privilèges implicites du pseudo-rôle public (USAGE sur les fonctions et les types).

- (3) **obj_typ** est une chaîne de caractères correspondant au type de l'objet. Valeurs acceptées : « table », « partitioned table » (sera assimilé à « table » et peut être écrit comme tel), « view », « materialized view », « foreign table », « sequence », « function », « aggregate », « type », « domain ».

Les valeurs de **obj_schema** s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'c_Bibliothèque' et non '"c_Bibliothèque"'.


C'est également vrai pour **obj_nom**, sauf dans le cas des **obj_typ** fonction et aggregate. Pour les **obj_typ** fonction et aggregate, les guillemets devront impérativement être présents dans **obj_nom** si le nom n'est pas normalisé – et seulement dans ce cas. Ainsi, on écrira 'suite_arithmetique(integer,integer)' (normalisé, sans guillemets), mais '"Suite Arithmétique"(integer,integer)' (non normalisé, avec guillemets).

Lorsque l'objet à déplacer est une fonction ou un agrégat, **obj_nom** doit spécifier les types de ses arguments, sous forme non abrégée (integer et pas int) et, s'il y en a plusieurs, sans espace après les virgules.

RÉSULTAT : « __ REINITIALISATION REUSSIE. » si la requête s'est exécutée normalement.

EXEMPLE. Le producteur du schéma c_bibliotheque a révoqué par erreur les droits de l'éditeur g_snum sur la table emprunt_livre. Il utilise la fonction asgard_intialise_obj pour les lui rendre sans risquer d'en oublier.

```
SELECT z_asgard.asgard_initialise_obj('c_bibliotheque', 'emprunt_livre',
'table') ;
```

Données	EXPLAIN	Messages	Notifications
 asgard_initialise_obj text			
1	__ REINITIALISATION REUSSIE.		

NOTICE: remise à zéro des privilèges manuels du pseudo-rôle public :

NOTICE: > néant

NOTICE: remise à zéro des privilèges des autres rôles (pour le producteur, les éventuels privilèges manquants sont réattribués) :

NOTICE: > REVOKE SELECT ON TABLE c_bibliotheque.emprunt_livre FROM g_consult

NOTICE: application des privilèges standards pour le rôle éditeur du schéma :

NOTICE: > GRANT SELECT, UPDATE, DELETE, INSERT ON TABLE c_bibliotheque.emprunt_livre TO g_snum

NOTICE: application des privilèges standards pour le rôle lecteur du schéma :

NOTICE: > GRANT SELECT ON TABLE c_bibliotheque.emprunt_livre TO g_consult

IV.5 — ASGARD_NETTOYAGE_ROLES

```
z_asgard.asgard_nettoyage_roles()
```

La fonction `asgard_nettoyage_roles` active la mise à jour des noms des rôles désignés dans la table de gestion comme producteurs, éditeurs et lecteurs, pour prendre en compte les changements de nom ou suppressions qui auraient pu avoir eu lieu :

- ◆ lorsque le nom d'un rôle a été modifié, l'ancien nom est remplacé par le nouveau ;
- ◆ lorsqu'un rôle n'existe plus, le champ lecteur ou éditeur est remis à NULL.

Pour plus de précisions, cf. III.10.

Dans la mesure où ne seront nettoyés que les enregistrements auxquels l'utilisateur a accès dans la vue `gestion_schema_usr`, il est fortement conseillé que cette fonction soit lancée par un membre de `g_admin`.

SCHÉMA : `z_asgard`.

SYNTAXE :

```
SELECT z_asgard.asgard_nettoyage_roles() ;
```

ARGUMENTS : néant.

RÉSULTAT : « __ NETTOYAGE REUSSI. » si la requête s'est exécutée normalement.

EXEMPLE. L'administrateur avait désigné le rôle de groupe `g_ped` comme producteur ou éditeur sur plusieurs schémas de la base. Ce rôle était nommé d'après le pôle « études et données », qui vient de changer de nom, devenant le pôle « connaissance des territoires ». Il décide de renommer le rôle de groupe en conséquence :

```
ALTER ROLE g_ped RENAME TO g_pct ;
```

Pour que ce changement devienne apparent dans la table de gestion, il exécute la fonction `asgard_nettoyage_roles`.

```
SELECT z_asgard.asgard_nettoyage_roles() ;
```

Données	EXPLAIN	Messages	Notifications
<div> <div>asgard_nettoyage_roles</div> <div>text</div> </div>			
1	__ NETTOYAGE REUSSI.		

NOTICE: [table de gestion] Schéma w_snow. Mise à jour du libellé du rôle producteur, renommé entre temps.
 DETAIL: Ancien nom "g_ped", nouveau nom "g_pct".
 NOTICE: [table de gestion] Schéma c_bibliotheque. Mise à jour du libellé du rôle éditeur, renommé entre temps.
 DETAIL: Ancien nom "g_ped", nouveau nom "g_pct".

IV.6 — ASGARD_SORTIE_GESTION_SCHEMA

```
z_asgard_admin.asgard_sortie_gestion_schema(n_schema text)
```

La fonction `asgard_sortie_gestion_schema` permet de supprimer de la table de gestion un schéma existant (qui échappera alors aux mécanismes de gestion des droits), en outrepassant les règles qui veulent que seules les lignes avec `creation` valant `False` puisse être ciblées par des commandes `DELETE` et que `creation` ne puisse être mis à `False` si le schéma existe.

Cf. aussi VI.7 pour une discussion sur les circonstances d'utilisation de cette fonction.

À noter que le déréférencement n'agit pas sur les droits. En particulier, les privilèges des éventuels ex-lecteur et ex-éditeur du schéma sont conservés.

Un schéma identifié comme appartenant à la nomenclature ne pourra pas être déréférencé (cf. III.2.B).

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

```
SELECT z_asgard_admin.asgard_sortie_gestion_schema(n_schema) ;
```


Ou, en nommant l'argument :

```
SELECT z_asgard_admin.asgard_sortie_gestion_schema(n_schema :=
n_schema) ;
```

ARGUMENT : *n_schema* est une chaîne de caractères correspondant au nom d'un schéma présumé référencé dans la table de gestion (sinon la fonction n'aura pas d'effet).

Les valeurs de *n_schema* s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'c_Bibliothèque' et non '"c_Bibliothèque"'.

RÉSULTAT : « __ DEREFERENCEMENT REUSSI. » si la requête s'est exécutée normalement.

EXEMPLE. L'administrateur ne souhaite plus que les droits sur le schéma special soit gérés par ASGARD. Il exécute la fonction asgard_sortie_gestion_schema :

```
SELECT z_asgard_admin.asgard_sortie_gestion_schema('special') ;
```

Données	EXPLAIN	Messages	Notifications
<div> <div>asgard_sortie_gestion_schema</div> <div>text</div> </div>			
1	__ DEREFERENCEMENT REUSSI.		

IV.7 —

ASGARD_INITIALISATION_GESTION_SCHEMA

```
z_asgard_admin.asgard_initialisation_gestion_schema([ exceptions text[]
DEFAULT NULL::text[] [, b_gs boolean DEFAULT false ] ] )
```

La fonction asgard_initialisation_gestion_schema intègre à la table de gestion des droits l'ensemble des schémas existants dans la base, hors schémas système et ceux qui sont (optionnellement) listés en argument.

Cf. VI.4 pour plus de précisions sur l'initialisation d'ASGARD après installation.

Si les schémas contiennent des objets, ASGARD mettra automatiquement en cohérence leurs propriétaires avec le celui du schéma. À noter que la fonction peut échouer si g_admin ne dispose pas de permissions suffisantes sur un schéma ou un objet, par

exemple si un super-utilisateur en est propriétaire. Dans ce cas aucun schéma n'est référencé.

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard_admin.asgard_initialisation_gestion_schema( [ exceptions  
[, b_gs ] ] ) ;
```

Les arguments doivent être spécifiés dans l'ordre.

Les arguments optionnels peuvent être omis, mais seulement s'ils sont à la fin de la liste.

Avec des arguments nommés :

```
SELECT z_asgard_admin.asgard_initialisation_gestion_schema( [  
    exceptions := exceptions [,  
    b_gs := b_gs ]]  
) ;
```

L'ordre des arguments nommés n'a pas d'importance.

Les arguments optionnels peuvent être omis.

Il est bien entendu possible de commencer avec des arguments positionnels (dans l'ordre) et terminer par des arguments nommés.

ARGUMENTS :

- (1) [optionnel] *exceptions* est un tableau de chaînes de caractères contenant les noms des schémas à omettre, le cas échéant (ex : `ARRAY['exception_1', 'exception_2']`). Les noms des schémas s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer `ARRAY['c_Bibliothèque']` et non `ARRAY['"c_Bibliothèque"]` ;
- (2) [optionnel] *b_gs* est un booléen indiquant si, dans l'hypothèse où un schéma serait déjà référencé – nécessairement comme non créé – dans la table de gestion, c'est le propriétaire du schéma qui doit devenir le producteur du schéma (`False`) ou le producteur pré-renseigné dans la table de gestion qui doit devenir le propriétaire du schéma (`True`). `False` par défaut.

RÉSULTAT : « __ FIN INITIALISATION. » si la requête s'est exécutée normalement.

EXEMPLE. Initialisation de la table de gestion sur une base contenant cinq schémas : `c_bibliotheque`, les deux schémas d'ASGARD et deux schémas `special_1` et `special_2` que l'administrateur ne souhaite pas gérer dans ASGARD pour le moment.

```
SELECT z_asgard_admin.asgard_initialisation_gestion_schema(
    ARRAY['special_1', 'special_2']
) ;
```

Données	EXPLAIN	Messages	Notifications
<div> <div>asgard_initialisation_gestion_schema</div> <div>text</div> </div>			
1	__ FIN INITIALISATION.		

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma `c_bibliotheque` (c).

NOTICE: (schéma `c_bibliotheque` pré-existant)

NOTICE: ... Schéma `c_bibliotheque` enregistré dans la table de gestion.

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma `z_asgard_admin` (z).

NOTICE: (schéma `z_asgard_admin` pré-existant)

NOTICE: ... Schéma `z_asgard_admin` enregistré dans la table de gestion.

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma `z_asgard` (z).

NOTICE: (schéma `z_asgard` pré-existant)

NOTICE: ... Schéma `z_asgard` enregistré dans la table de gestion.

Si l'administrateur avait voulu référencer la totalité des schémas existants de sa base, il aurait simplement lancé :

```
SELECT z_asgard_admin.asgard_initialisation_gestion_schema() ;
```

IV.8 —

ASGARD_ALL_LOGIN_GRANT_ROLE

```
z_asgard_admin.asgard_all_login_grant_role(n_role text [, b boolean
DEFAULT true ] )
```

La fonction `asgard_all_login_grant_role` confère à tous les rôles de connexion du serveur l'appartenance au rôle donné en argument (s'ils n'en étaient pas déjà membres).

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard_admin.asgard_all_login_grant_role(n_role [, b ] ) ;
```

Les arguments doivent être spécifiés dans l'ordre.
L'argument optionnel peut être omis.

Avec des arguments nommés :

```
SELECT z_asgard_admin.asgard_all_login_grant_role(  
    n_role := n_role [,  
    b := b ]  
    ) ;
```

L'ordre des arguments nommés n'a pas d'importance.
L'argument optionnel peut être omis.

ARGUMENTS :

- (1) *n_role* est une chaîne de caractères présumée correspondre à un nom de rôle valide. Les valeurs de *n_role* s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'g_Rôle' et non '"g_Rôle"' ;
- (2) [optionnel] *b* est un booléen. Si ce paramètre prend la valeur False et qu'un rôle de connexion est déjà membre du rôle considéré par héritage³⁷, la fonction ne fait rien. S'il vaut True (défaut), la fonction ne passera un rôle de connexion que s'il est lui-même membre du rôle considéré.

RÉSULTAT : un entier correspondant au nombre de rôles pour lesquels la permission a été accordée. Les commandes GRANT effectivement exécutées apparaissent dans l'onglet « Messages » de l'éditeur de requêtes.

EXEMPLE. Après avoir installé ASGARD, l'administrateur veut rendre tous les rôles de connexion qu'il avait précédemment créés membres de g_consult, comme préconisé dans la partie II.3.A.

³⁷ Par exemple, si jon.snow est membre de g_snum, qui est membre de g_consult, mais que jon.snow lui-même n'est pas directement membre du rôle g_consult, alors la version asgard_all_login_grant_role('g_consult') confère g_consult à jon.snow et pas la version asgard_all_login_grant_role('g_consult', False).

```
SELECT z_asgard_admin.asgard_all_login_grant_role('g_consult') ;
```

Données	EXPLAIN	Messages	Notifications
<div> <div>asgard_all_login_grant_role</div> <div>integer</div> </div>			
1		5	

```
NOTICE: > GRANT g_consult TO "michel.chrestien"
NOTICE: > GRANT g_consult TO "felix.grandet"
NOTICE: > GRANT g_consult TO "esther.gobseck"
NOTICE: > GRANT g_consult TO "etienne.lousteau"
NOTICE: > GRANT g_consult TO "eugenie.grandet"
```

IV.9 — ASGARD_IMPORT_NOMENCLATURE

```
z_asgard_admin.asgard_import_nomenclature( [ domaines text[] DEFAULT
NULL::text[] ] )
```

La fonction `asgard_import_nomenclature` importe dans la table de gestion les schémas manquants de la nomenclature nationale – ou de certains domaines de la nomenclature qui sont alors listés en argument.

Le rôle `g_admin` est désigné comme producteur, le rôle `g_consult` comme lecteur.

On notera que la fonction ne crée pas les schémas, elle se contente de les référencer dans la table de gestion d'ASGARD (creation vaut `False`) en vue d'une création future.

`asgard_import_nomenclature` ne fait aucune différence si un schéma de la nomenclature a un homonyme parmi les schémas existants de la base. Elle l'importera dans la table de gestion sous la même forme que les autres : il ne sera pas marqué comme créé et le schéma existant ne sera pas affecté par les modifications faites dans la table de gestion tant qu'il n'aura pas été explicitement référencé par l'administrateur, soit via les fonctions de référencement usuelles (`asgard_initialisation_gestion_schema` et `asgard_initialise_schema`), soit simplement en basculant `creation` sur `True`.

`asgard_import_nomenclature` pourra notamment servir au moment de l'initialisation d'ASGARD après installation (cf. VI.4 pour plus de précisions), mais elle peut être lancée ou relancée à tout moment pour importer des domaines supplémentaires, voire tous les schémas restants de la nomenclature.

Lorsque `asgard_import_nomenclature` est ré-exécutée sur des sections de la nomenclature qui avaient déjà été importées dans la table de gestion, les champs `nomenclature`, `niv1`, `niv1_abr`, `niv2`, `niv2_abr` sont réinitialisés selon la source.

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

```
SELECT z_asgard_admin.asgard_import_nomenclature( [ domaines ] ) ;
```

Ou, en nommant l'argument :

```
SELECT z_asgard_admin.asgard_import_nomenclature( [ domaines :=  
domaines ] ) ;
```

L'argument, optionnel, peut être omis.

ARGUMENT : [optionnel] *domaines* est un tableau de chaînes de caractères (`text[]`) contenant les noms des domaines à importer, soit le « niveau 1 » de la nomenclature des schémas (repris dans les champs `niv1/niv1_abr` de la table de gestion). Si aucun argument n'est fourni, toute la nomenclature est importée hors schémas déjà référencés et/ou existants.

RÉSULTAT : « __ FIN IMPORT NOMENCLATURE. » si la requête s'est exécutée normalement.

Des messages informent l'opérateur des schémas effectivement ajoutés à la table de gestion ou pour lesquels les champs relatifs à la nomenclature ont été modifiés.

Les domaines peuvent être renseignés indifféremment sous leur forme littérale (telle qu'elle apparaît dans le champ `niv1`) ou abrégée (comme dans `niv1_abr`).

Liste des domaines (par ordre alphabétique) :

agriculture	Agriculture
air_climat	Air & climat
amenagement_urbanisme	Aménagement & urbanisme
culture_societe_service	Culture, société & services
donnees_generique	Données génériques
eau	Eau
foncier_sol	Foncier & sol
foret	Forêt
habitat_politique_de_la_ville	Habitat & politique de la ville
mer_littoral	Mer & littoral
nature_paysage_biodiversite	Nature, paysage & biodiversité
nuisance	Nuisances
reseau_energie_divers	Réseaux & énergie
risque	Risques
site_industriel_production	Sites industriels & production
socio_economie	Socio-économie
transport_deplacement	Déplacements
transport_infrastructure	Infrastructures de transport

EXEMPLE. Pour importer les schémas des domaines « transport_deplacement » et « transport_infrastructure » :

```
SELECT z_asgard_admin. asgard_import_nomenclature(
    ARRAY['transport_deplacement', 'Infrastructures de transport']
) ;
```

Données	EXPLAIN	Messages	Notifications
asgard_import_nomenclature text			
1	__ FIN IMPORT NOMENCLATURE.		

NOTICE: Le schéma c_tr_depl_securite_routiere a été ajouté à la table de gestion.

NOTICE: Le schéma c_tr_depl_collectif a été ajouté à la table de gestion.

NOTICE: Le schéma c_tr_depl_exceptionnel a été ajouté à la table de gestion.

NOTICE: Le schéma c_tr_depl_marchandise a été ajouté à la table de gestion.

NOTICE: Le schéma c_tr_depl_mat_dangereuse a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_depl_trafic a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_aerien a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_circulation_douce a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_ferroviaire a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_fluvial a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_maritime a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_plateforme_multimod a été ajouté à la table de gestion.
 NOTICE: Le schéma c_tr_infra_routier a été ajouté à la table de gestion.

EXEMPLE. Pour importer la totalité de la nomenclature :

```
SELECT z_asgard_admin. asgard_import_nomenclature() ;
```

Données	EXPLAIN	Messages	Notifications
		asgard_import_nomenclature text	
1		__ FIN IMPORT NOMENCLATURE.	

Tous les schémas de la nomenclature sont listés dans l'onglet Messages, sous la même forme que dans l'exemple précédent.

IV.10 — ASGARD_REAFFECTE_ROLE

```
z_asgard_admin.asgard_reaffecte_role(n_role text [, n_role_cible text  

    DEFAULT NULL::text [, b_hors_asgard boolean DEFAULT false [, b_privileges  

    boolean DEFAULT true [, b_default_acl boolean DEFAULT false ]]]) )
```

asgard_reaffecte_role transfère les droits sur les objets d'un rôle à un autre, incluant s'il y a lieu ses fonctions de producteur, éditeur et lecteur. Si aucun rôle cible n'est spécifié, les privilèges sont simplement supprimés et, le cas échéant, g_admin reçoit la propriété des objets.

Il peut être nécessaire d'exécuter cette fonction en tant que super-utilisateur si certains droits ont été conférés par un super-utilisateur et/ou portent sur des objets appartenant à un super-utilisateur.

Selon le paramétrage, la fonction peut viser uniquement les objets des schémas référencés dans ASGARD ou la totalité des objets de la base, y compris les objets partagés entre les bases (bases, tablespaces). Elle peut également être paramétrée :

- ◆ pour transférer la propriété des objets à un rôle déterminé, mais pas les privilèges non liés à la propriété (indépendamment de la question des privilèges par défaut, qui sont exclusivement gérés par le paramètre ***b_default_acl***) ;
- ◆ pour s'appliquer sur les privilèges par défaut (définis par ALTER DEFAULT PRIVILEGES), en plus des privilèges effectifs sur les objets.

asgard_reaffecte_role n'a aucune action sur les permissions dont disposerait le rôle sur d'autres rôles ou inversement.

Le pseudo-rôle public n'est pas reconnu par cette fonction.

asgard_reaffecte_role est notamment utile pour préparer la suppression d'un rôle, dans la mesure où celle-ci n'est possible qu'à partir du moment où le rôle ne dispose plus d'aucun privilège sur les objets d'aucune base.

SCHÉMA : z_asgard_admin.

SYNTAXE :

Avec des arguments positionnels :

```
SELECT z_asgard_admin.asgard_reaffecte_role(n_role [, n_role_cible [,  
b_hors_asgard [, b_privileges [, b_default_acl ]]]] ) ;
```

Les arguments doivent être spécifiés dans l'ordre.

Les arguments optionnels peuvent être omis, mais seulement s'ils sont à la fin de la liste.

Avec des arguments nommés :

```
SELECT z_asgard_admin.asgard_reaffecte_role(  
    n_role := n_role [,  
    n_role_cible := n_role_cible [,  
    b_hors_asgard := b_hors_asgard [,  
    b_privileges := b_privileges [,  
    b_default_acl := b_default_acl ]]]]  
) ;
```

L'ordre des arguments nommés n'a pas d'importance.

Les arguments optionnels peuvent être omis.

Il est bien entendu possible de commencer avec des arguments positionnels (dans l'ordre) et terminer par des arguments nommés.

ARGUMENTS :

- (1) ***n_role*** est une chaîne de caractères correspondant au nom du rôle (présumé valide) dont les privilèges et propriétés doivent être transférées ;
- (2) [optionnel] ***n_role_cible*** est une chaîne de caractères correspondant au nom du rôle cible pour le transfert (présumé valide). Si NULL, ***g_admin*** reçoit la propriété éventuelle des objets et les privilèges sont révoqués sans transfert ;
- (3) [optionnel] ***b_hors_asgard*** est un booléen, valeur par défaut False. Si ce paramètre vaut True, la propriété et les privilèges sur les objets des schémas non gérés par ASGARD ou hors schémas sont pris en compte ;
- (4) [optionnel] ***b_privileges*** est un booléen, valeur par défaut True. Indique si, dans l'hypothèse où le rôle cible est spécifié, celui-ci doit recevoir les privilèges et propriétés du rôle (True) ou seulement ses propriétés (False) ;
- (5) [optionnel] ***b_default_acl*** est un booléen, valeur par défaut False. Indique si les privilèges par défaut doivent être pris en compte (True) ou non (False) en plus du reste. Si le paramètre ***b_hors_asgard*** vaut False, seuls les privilèges par défaut définis sur les schémas d'ASGARD seront considérés.

Les valeurs de ***n_role*** et ***n_role_cible*** s'écrivent sans les guillemets doubles nécessaires aux identifiants des commandes SQL, même s'il s'agit de noms non normalisés. Par exemple, on veillera bien à indiquer 'g_Rôle' et non '"g_Rôle"'.

RÉSULTAT : « __ REAFFECTATION REUSSIE » si la fonction s'est exécutée sans erreur.

Des messages rendent compte des commandes effectivement lancées.

On notera que, pour modifier les propriétaires des objets hors schémas référencés par ASGARD, la fonction utilise la commande REASSIGN OWNED. Celle-ci apparaît dans les messages, mais le détail des propriétés effectivement réaffectées n'est pas fourni. Il peut d'ailleurs n'y en avoir aucune si le rôle n'était propriétaire d'aucun objet de ce type.

EXEMPLE. On veut supprimer le rôle ***g_admin_delegate***, qui était éditeur de ***z_asgard*** et disposait du privilège CREATE sur la base, qui lui permettait de créer des schémas. Ce rôle était en outre producteur d'un schéma ***z_admin_delegate***, qui pour l'heure est vide.

Si on lance la fonction dans sa forme la plus simple, avec un seul argument :

```
SELECT z_asgard_admin.asgard_reaffecte_role('g_admin_delegate') ;
```

Données	EXPLAIN	Messages	Notifications
<div> <div>asgard_reaffecte_role</div> <div>text</div> </div>			
1	__ REAFFECTATION REUSSIE		

NOTICE: attribution de la propriété du schéma et des objets au rôle producteur du schéma z_admin_delegue :

NOTICE: > ALTER SCHEMA z_admin_delegue OWNER TO g_admin

NOTICE: ... Le producteur du schéma z_admin_delegue a été redéfini.

NOTICE: suppression des privilèges de l'ancien éditeur du schéma z_asgard :

NOTICE: > REVOKE USAGE ON SCHEMA z_asgard FROM g_admin_delegue

NOTICE: > REVOKE SELECT ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > REVOKE INSERT ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > REVOKE UPDATE ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > REVOKE DELETE ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > REVOKE SELECT ON TABLE z_asgard.gestion_schema_etr FROM g_admin_delegue

NOTICE: > REVOKE INSERT ON TABLE z_asgard.gestion_schema_etr FROM g_admin_delegue

NOTICE: > REVOKE UPDATE ON TABLE z_asgard.gestion_schema_etr FROM g_admin_delegue

NOTICE: > REVOKE DELETE ON TABLE z_asgard.gestion_schema_etr FROM g_admin_delegue

NOTICE: > REVOKE SELECT ON TABLE z_asgard.gestion_schema_usr FROM g_admin_delegue

NOTICE: > REVOKE INSERT ON TABLE z_asgard.gestion_schema_usr FROM g_admin_delegue

NOTICE: > REVOKE UPDATE ON TABLE z_asgard.gestion_schema_usr FROM g_admin_delegue

NOTICE: > REVOKE DELETE ON TABLE z_asgard.gestion_schema_usr FROM g_admin_delegue

NOTICE: ... L'éditeur du schéma z_asgard a été redéfini.

La propriété du schéma z_admin_delegue est conférée à g_admin, l'éditeur du schéma z_asgard est supprimé. Le paramétrage choisi a limité le champ d'action de la fonction aux seuls schémas référencés par z_asgard, ainsi le privilège CREATE sur la base est toujours présent, ce qui empêche la suppression du rôle.

On aurait pu prendre en considération la totalité des objets de la base (et les objets partagés entre bases) avec la forme suivante :

```
SELECT z_asgard_admin.asgard_reaffecte_role('g_admin_delegue',
b_hors_asgard := True) ;
```

En plus des messages déjà précédemment renvoyés, on aurait alors :

NOTICE: > REASSIGN OWNED BY g_admin_delegue TO g_admin

NOTICE: ... Le cas échéant, la propriété des objets hors schémas référencés par ASGARD a été réaffectée.

NOTICE: > REVOKE CREATE ON DATABASE geobase_snum FROM g_admin_delegue

NOTICE: ... Les privilèges résiduels du rôle g_admin_delegue sur les objets hors schémas ont été révoqués.

Cette fois, le privilège CREATE sur la base est bien révoqué. Le rôle n'a plus aucun droit sur les objets et peut donc être supprimé.

S'il est possible que des privilèges par défaut aient été définis pour g_admin_delegue, alors la forme suivante aurait permis de les nettoyer également :

```
SELECT z_asgard_admin.asgard_reaffecte_role('g_admin_delegue',
b_hors_asgard := True, b_default_acl := True) ;
```

[idem]

NOTICE: > ALTER DEFAULT PRIVILEGES FOR ROLE g_admin IN SCHEMA z_admin REVOKE SELECT ON TABLES FROM g_admin_delegue

NOTICE: ... Les privilèges par défaut du rôle g_admin_delegue ont été supprimés.

Il était également possible de transférer les droits de g_admin_delegue à un autre rôle plutôt que de les supprimer. Dans cet exemple, ce rôle cible est g_snum.

```
SELECT z_asgard_admin.asgard_reaffecte_role('g_admin_delegue', 'g_snum',
b_hors_asgard := True) ;
```

Données	EXPLAIN	Messages	Notifications
asgard_reaffecte_role text			
1	__ REAFFECTATION REUSSIE		

NOTICE: attribution de la propriété du schéma et des objets au rôle producteur du schéma z_admin_delegue :

NOTICE: > ALTER SCHEMA z_admin_delegue OWNER TO g_snum

NOTICE: ... Le producteur du schéma z_admin_delegue a été redéfini.

NOTICE: suppression et transfert vers le nouvel éditeur des privilèges de l'ancien éditeur du schéma z_asgard :

NOTICE: > GRANT USAGE ON SCHEMA z_asgard TO g_snum

NOTICE: > REVOKE USAGE ON SCHEMA z_asgard FROM g_admin_delegue

NOTICE: > GRANT SELECT ON TABLE z_asgard.qgis_menubuilder_metadata TO g_snum

NOTICE: > REVOKE SELECT ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > GRANT INSERT ON TABLE z_asgard.qgis_menubuilder_metadata TO g_snum

NOTICE: > REVOKE INSERT ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > GRANT UPDATE ON TABLE z_asgard.qgis_menubuilder_metadata TO g_snum

NOTICE: > REVOKE UPDATE ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

NOTICE: > GRANT DELETE ON TABLE z_asgard.qgis_menubuilder_metadata TO g_snum

NOTICE: > REVOKE DELETE ON TABLE z_asgard.qgis_menubuilder_metadata FROM g_admin_delegue

[... idem pour les vues gestion_schema_etr et gestion_schema_use]

NOTICE: ... L'éditeur du schéma z_asgard a été redéfini.

NOTICE: > REASSIGN OWNED BY g_admin_delegue TO g_snum

NOTICE: ... Le cas échéant, la propriété des objets hors schémas référencés par ASGARD a été réaffectée.

NOTICE: > REVOKE CREATE ON DATABASE geobase_dev FROM g_admin_delegue

NOTICE: > GRANT CREATE ON DATABASE geobase_dev TO g_snum

NOTICE: ... Les privilèges résiduels du rôle g_admin_delegue sur les objets hors schémas ont été réaffectés.

IV.11 — ASGARD_INITIALISE_ALL_SCHEMAS

```
z_asgard_admin.asgard_initialise_all_schemas( [ variante integer DEFAULT
0 ] )
```

`asgard_initialise_all_schemas` est une fonction de premiers secours. Selon la variante utilisée, elle permet :

- ◆ de réinitialiser les droits sur tous les schémas référencés en une seule commande. En substance, il s'agit d'une application massive d'`asgard_initialise_schema` ;
- ◆ de vérifier la cohérence des propriétaires des objets avec ceux des schémas (référéncés) qui les contiennent. Cette fonctionnalité devra notamment être mise à profit en cas de désactivation malencontreuse des déclencheurs sur événement d'ASGARD ;
- ◆ de remettre à plat propriétaires et privilèges sur les schémas d'ASGARD – par exemple en cas de modification accidentelle des droits. L'effet est alors presque identique à l'application de la fonction `asgard_initialise_schema` sur `z_asgard` et `z_asgard_admin`, si ce qu' `asgard_initialise_all_schemas` saura corriger la perte du privilège `SELECT` de `g_admin_ext` sur `gestion_schema`, au contraire de `asgard_initialise_schema`.

Il peut être nécessaire d'exécuter la fonction avec un rôle super-utilisateur si certains schémas référencés (ou objets qui s'y trouvent) appartiennent à des super-utilisateurs.

Contrairement à la plupart des fonctions d'ASGARD, `asgard_initialise_all_schemas` ne retourne pas d'erreur si elle n'est pas en mesure de traiter un schéma. Elle poursuit sur les autres et informe simplement l'utilisateur du problème rencontré.

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

```
SELECT z_asgard_admin.asgard_initialise_all_schemas ( [ variante ] ) ;
```

Ou, en nommant l'argument :

```
SELECT z_asgard_admin.asgard_initialise_all_schemas ( [ variante :=
variante ] ) ;
```

L'argument, optionnel, peut être omis.

Tout particulièrement dans sa variante 0, et dans une moindre mesure pour la variante 1, la fonction `asgard_initialise_all_schemas` peut avoir un temps d'exécution conséquent sur les bases qui comportent un grand nombre de schémas et/ou d'objets, de l'ordre de plusieurs minutes voire dizaines de minutes.

Pour améliorer la performance, il est recommandé d'exécuter simultanément une commande SET qui bloque temporairement l'émission des messages par lesquels ASGARD rend compte de toutes les opérations réalisées.

```
SET LOCAL client_min_messages = 'ERROR' ;  
SELECT z_asgard_admin.asgard_initialise_all_schemas( [ variante ] ) ;
```

ARGUMENT : [optionnel] *variante* est un entier compris entre 0 et 2 indiquant la variante à utiliser (cf. ci-après). 0 par défaut.

RÉSULTAT : NULL si la requête s'est exécutée normalement, sinon la liste des schémas qui n'ont pas pu être traités. Se reporter dans ce cas à l'onglet Messages pour le détail des erreurs.

Plus généralement, des messages informent l'utilisateur de toutes les opérations réalisées.

VARIANTES :

- 0 La fonction s'applique à tous les schémas référencés + `z_asgard` et `z_asgard_admin`, qu'ils soient référencés ou non. Elle contrôle la cohérence des propriétaires des objets et du schéma avec le producteur désigné et réinitialise les droits selon les privilèges standards du producteur, de l'éditeur et du lecteur renseignés dans la table de gestion d'ASGARD (ceux des autres rôles sont révoqués, ainsi que les éventuels privilèges par défaut portant sur le schéma). Sur les schémas `z_asgard` et `z_asgard_admin`, elle exécute les mêmes opérations que la variante 2.
- 1 La fonction s'applique à tous les schémas référencés + `z_asgard` et `z_asgard_admin`, qu'ils soient référencés ou non. Elle contrôle uniquement la cohérence des propriétaires des objets et du schéma avec le producteur désigné dans la table de gestion.
- 2 La fonction s'applique exclusivement sur les schémas `z_asgard` et `z_asgard_admin`, et ce même s'ils n'étaient pas référencés dans ASGARD (auquel cas la fonction les référence). Elle contrôle les propriétaires, efface les modifications manuelles des droits et rétablit les privilèges nécessaires au bon fonctionnement d'ASGARD. Le cas échéant, elle applique les privilèges standards de l'éditeur et du lecteur déclarés dans la table de gestion.

EXEMPLE. Nettoyage complet des droits après l'initialisation sur une base existante.

```
SELECT z_asgard_admin.asgard_initialise_all_schemas() ;
```

Cette commande équivaut à :

```
SELECT z_asgard_admin.asgard_initialise_all_schemas(0) ;
```

EXEMPLE. Restauration des droits sur z_asgard_admin après une suppression manuelle malencontreuse des privilèges (absolument essentiels) de g_admin_ext.

```
SELECT z_asgard_admin.asgard_initialise_all_schemas(2) ;
```

```
NOTICE: (temporaire) droits a minima pour le propriétaire de la vue gestion_schema_usr :
NOTICE: > GRANT USAGE ON SCHEMA z_asgard_admin TO g_admin_ext
NOTICE: > GRANT SELECT ON TABLE z_asgard_admin.gestion_schema TO g_admin_ext
NOTICE: -----
NOTICE: (ré)attribution de la propriété des objets au rôle producteur du schéma :
NOTICE: > néant
NOTICE: remise à zéro des privilèges manuels du pseudo-rôle public :
NOTICE: > néant
NOTICE: remise à zéro des privilèges des autres rôles (pour le producteur, les éventuels privilèges
manquants sont réattribués) :
NOTICE: > REVOKE USAGE ON SCHEMA z_asgard_admin FROM g_admin_ext
NOTICE: > REVOKE SELECT ON TABLE z_asgard_admin.gestion_schema FROM g_admin_ext
NOTICE: rétablissement des privilèges attendus pour g_admin_ext :
NOTICE: > GRANT USAGE ON SCHEMA z_asgard_admin TO g_admin_ext
NOTICE: > GRANT INSERT, SELECT, UPDATE, DELETE ON TABLE z_asgard_admin.gestion_schema TO
g_admin_ext
NOTICE: > GRANT SELECT ON TABLE z_asgard_admin.asgard_parametre TO g_admin_ext
NOTICE: suppression des privilèges par défaut :
NOTICE: > néant
NOTICE: ... Le schéma z_asgard_admin a été traité
NOTICE: -----
```

(les messages relatifs à z_asgard sont omis)

Après cette commande ASGARD sera de nouveau fonctionnel.

IV.12 — ASGARD_DIAGNOSTIC

```
z_asgard_admin.asgard_diagnostic()
```

asgard_diagnostic permet de contrôler rapidement que tout est en ordre concernant les droits sur les schémas référencés par ASGARD et leur contenu.

Elle renvoie à l'utilisateur une liste d'« anomalies », c'est-à-dire de points sur lesquels les droits effectifs s'écartent du standard d'ASGARD. Ces anomalies peuvent correspondre à des personnalisations de droits sciemment réalisées par l'ADL et ne sont donc pas nécessairement problématiques. Certaines peuvent toutefois causer des dysfonctionnements importants si elles ne sont pas corrigées au plus tôt. La fonction qualifie la gravité des anomalies grâce à un attribut, critique, évoqué ci-après.

Concrètement, la fonction `asgard_diagnostic` considère comme des anomalies :

- ◆ les droits manquants pour les rôles `g_consult` et `g_admin_ext` sur les objets des schémas d'ASGARD³⁸ (critique) ;
- ◆ les objets dont le propriétaire n'est pas le producteur identifié pour leur schéma (critique) ;
- ◆ les privilèges manquants du propriétaire/producteur, de l'éditeur et du lecteur au regard des droits prévus par ASGARD pour chacun d'eux (non critique) ;
- ◆ les privilèges alloués à d'autres rôles que les propriétaire/producteur, éditeur et lecteur, ou conférés à ces derniers mais outrepassant les droits normaux (non critique) ;
- ◆ les fonctions sur lesquelles le pseudo-rôle public n'a pas le privilège `EXECUTE` et les types sur lesquels il n'a pas le privilège `USAGE` (droits accordés automatiquement par PostgreSQL et qu'ASGARD ne remet pas en cause, non critique) ;
- ◆ tous les privilèges par défaut, leur usage n'étant pas prévu par ASGARD (non critique).

La fonction `asgard_diagnostic` peut avoir un temps d'exécution conséquent sur les bases qui comportent un grand nombre de schémas et/ou d'objets, de l'ordre de plusieurs minutes.

SCHÉMA : `z_asgard_admin`.

SYNTAXE :

La syntaxe d'`asgard_diagnostic` est légèrement différente de celle des autres fonctions, dans la mesure où son résultat est une table avec plusieurs attributs :

```
SELECT { * | expression [, ...] }  
FROM z_asgard_admin.asgard_diagnostic() ;
```

38 En supposant que `z_asgard` et `z_asgard_admin` aient été référencés.

expression est un appel aux attributs du résultat de la fonction, sinon * pour tous les attributs

ARGUMENT : néant.

RÉSULTAT : une table avec autant d'enregistrements que d'anomalies identifiées et cinq attributs :

- ◆ `nom_schema` est le nom du schéma concerné.
- ◆ `nom_objet` est le nom de l'objet concerné. NULL pour un privilège par défaut. Pour un attribut, `nom_objet` est formé du nom de la table ou assimilé, suivi du nom de l'attribut entre parenthèses – par exemple, « `table_a (id)` ».
- ◆ `typ_objet` est le type d'objet concerné (écrit en français sous une forme lisible non standardisée).
- ◆ `critique` est un booléen qui qualifie la gravité de l'anomalie constatée. Lorsqu'il vaut True, il est recommandé de remédier immédiatement au problème, qui est susceptible d'affecter le fonctionnement d'ASGARD. Lorsqu'il vaut False, l'anomalie met simplement en évidence une répartition des droits qui n'est pas celle qu'ASGARD met en œuvre par défaut. Il revient alors à l'ADL d'évaluer la légitimité de cette personnalisation.
- ◆ `anomalie` est la description de l'anomalie.

Les anomalies peuvent être corrigées par des commandes SQL standard (par exemple GRANT) ou à l'aide des fonctions de réinitialisation des droits d'ASGARD :

- ◆ `asgard_initialise_obj` pour réinitialiser les droits sur un objet donné ;
- ◆ `asgard_initialise_schema` pour réinitialiser les droits sur un schéma et les objets qu'il contient ;
- ◆ `asgard_initialise_all_schemas` pour réinitialiser les droits sur tous les schémas référencés et les objets qu'ils contiennent.

Les fonctions ci-avant pourront remédier à toutes les catégories d'anomalies identifiées par `asgard_diagnostic` sauf les privilèges manquants du pseudo-rôle public sur les fonctions et les types, volontairement exclus du champ des mécanismes automatisés d'ASGARD et qui, s'il y a lieu, devront être rétablis par des commandes GRANT manuelles.

`asgard_diagnostic` se veut par ailleurs exhaustive. Elle considère tous les types d'objets rattachés à des schémas, y compris ceux qui ne sont d'ordinaire pas pris en charge par les mécanismes automatiques et les fonctions d'ASGARD (cf. III.9). Dans ce cas, les corrections éventuelles devront être réalisées manuellement par l'ADL, les fonctions de réinitialisation étant inopérantes.

asgard_diagnostic est également susceptible d'afficher des objets qui restent habituellement invisibles aux yeux des utilisateurs.

EXEMPLE. On se place dans le cas très hypothétique où, suite à la désactivation accidentelle du déclencheur sur évènement qui veille à ce que ce type de situation ne se produise jamais (cf. II.1), le propriétaire de la table `table_13` a pu être modifié via la commande suivante sans qu'ASGARD n'annule sur le champ l'opération :

```
ALTER TABLE c_agri_agroalimentaire.table_13 OWNER TO g_admin_ext ;
```

Cette table se trouve ainsi avoir un propriétaire (`g_admin_ext`) qui n'est pas le producteur du schéma qui la contient (`postgres`).

Lorsque l'ADL lance la fonction de diagnostic, elle lui renvoie le résultat suivant :

nom_schema text	nom_objet text	typ_objet text	critique boolean	anomalie text
c_agri_agroalimentaire	table_13	table	true	le propriétaire (g_admin_ext) n'est pas le producteur désigné pour le schéma (postgres)
c_agri_agroalimentaire	table_13_id_seq	séquence	true	le propriétaire (g_admin_ext) n'est pas le producteur désigné pour le schéma (postgres)
c_agri_agroalimentaire	table_13	type	true	le propriétaire (g_admin_ext) n'est pas le producteur désigné pour le schéma (postgres)
c_agri_agroalimentaire	_table_13	type	true	le propriétaire (g_admin_ext) n'est pas le producteur désigné pour le schéma (postgres)
c_agri_agroalimentaire	table_13_id_seq	type	true	le propriétaire (g_admin_ext) n'est pas le producteur désigné pour le schéma (postgres)

Outre la table elle-même, le résultat recense une séquence – utilisée pour la clé primaire de la table – et trois types qui sont générés automatiquement par PostgreSQL à la création de la table ou de la séquence associée. Pour tous ces objets, le propriétaire est hérité du propriétaire de la table dont ils dépendent. L'ADL n'a donc pas à s'en préoccuper, il lui suffit de corriger le propriétaire de la table.

Il pourra utiliser pour ce faire la fonction `asgard_initialise_obj`.

```
SELECT z_asgard.asgard_initialise_obj('c_agri_agroalimentaire',
'table_13', 'table') ;
```

NOTICE: réattribution de la propriété de table_13 au rôle producteur du schéma :

NOTICE: > ALTER table c_agri_agroalimentaire.table_13 OWNER TO postgres

NOTICE: remise à zéro des privilèges manuels du pseudo-rôle public :

NOTICE: > néant

NOTICE: remise à zéro des privilèges des autres rôles (pour le producteur, les éventuels privilèges manquants sont réattribués) :

NOTICE: > néant

Si l'ADL exécute de nouveau la fonction de diagnostic, il constatera qu'elle n'identifie plus d'anomalie.

V — MENU POUR QGIS

V.1 — PRINCIPE

Pour générer sous QGIS un menu d'accès aux données stockées dans la base PostgreSQL, ASGARD s'appuie sur l'extension QGIS AsgardMenu, dérivée de l'extension MenuBuilder développée initialement par la société Oslandia pour la DREAL ARA³⁹.

AsgardMenu est disponible sur le dépôt MTE⁴⁰. Il s'agit pour l'heure d'une version alpha, adaptée a minima pour fonctionner avec ASGARD, qui sera perfectionnée dans les prochains mois.

ASGARD crée une vue spécifique pour AsgardMenu, qui construit dynamiquement un profil « consultation » variable selon le rôle utilisé au lancement de QGIS pour la connexion à la base PostgreSQL. Concrètement, un utilisateur trouvera dans son menu toutes les tables et vues sur lesquelles il dispose au moins d'un accès en lecture et qui sont localisées dans des schémas référencés par ASGARD.

L'arborescence du menu est basée sur les champs niv1 et niv2 de la table de gestion. Ainsi, pour une table (ou vue) donnée :

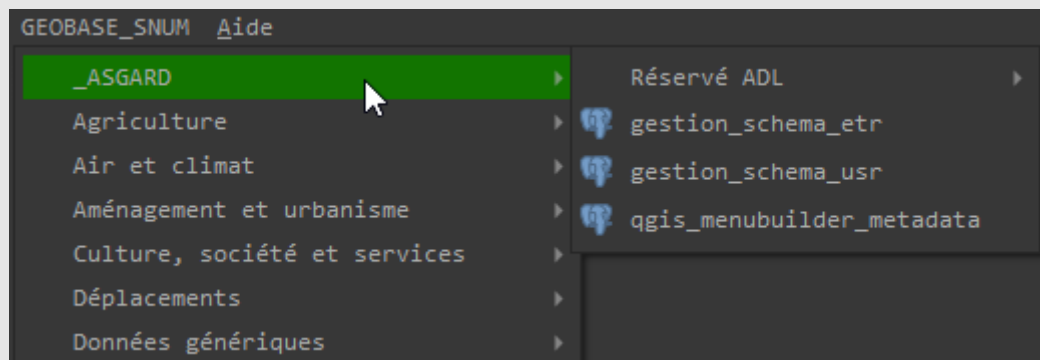
- ◆ le niveau zéro de l'arborescence (nom qui apparaît dans la barre des menus) est le nom de la base ;
- ◆ le premier niveau de l'arborescence est niv1 ou, si niv1 est NULL, le nom du schéma contenant la table ;
- ◆ le second niveau de l'arborescence est niv2 ou, si niv2 est NULL, la table elle-même ;

39 Copyright Oslandia – <http://www.oslandia.com>. MenuBuilder est publié sous licence GPL-2.0 ou postérieure sur le dépôt des extensions QGIS – <http://plugins.qgis.org/plugins/MenuBuilder/> – et le GitLab de la société Oslandia – <https://gitlab.com/Oslandia/qgis/qgis-menu-builder>.

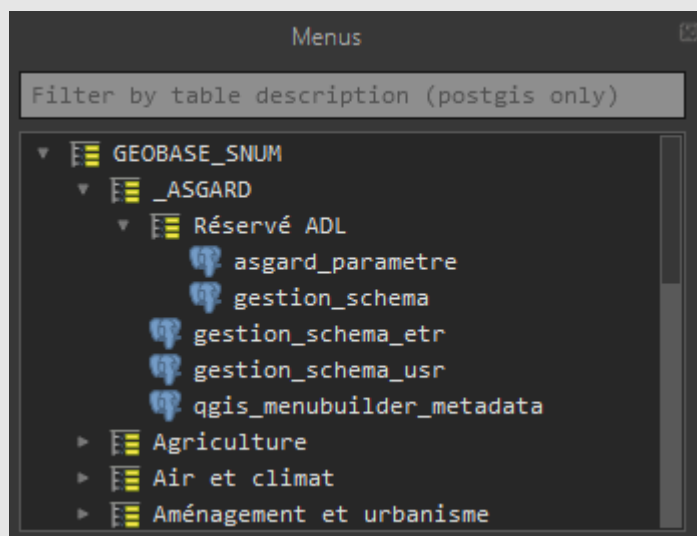
40 Copyright République Française, 2020. AsgardMenu est publié sous licence GPL-3.0. Dépôt : <http://piece-jointe-carto.developpement-durable.gouv.fr/NAT002/QGIS/plugins/plugins.xml>.

- ◆ si niv2 était renseigné, le troisième niveau d'arborescence est la table.

EXEMPLE. Avec un menu situé dans la barre de menus :



Avec un menu situé dans un panneau (apporte une fonctionnalité supplémentaire de filtre par recherche dans le commentaire de la table) :

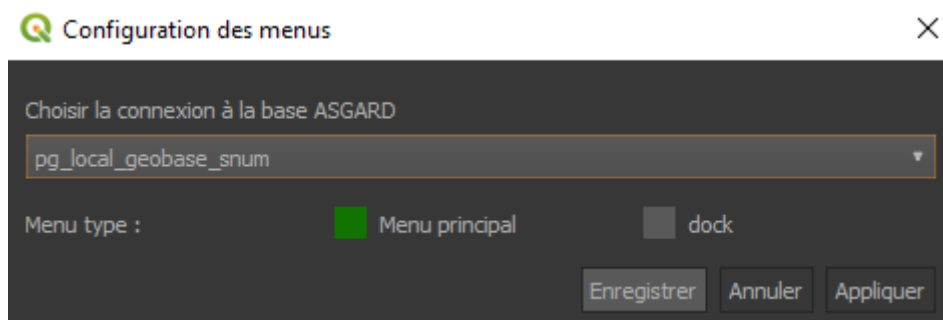


La vue en question, `qgis_menubuilder_metadata`, se trouve dans le schéma `z_asgard`. Elle est visible par tous les rôles membres de `g_consult`⁴¹.

⁴¹ D'où la nécessité de s'assurer que les rôles de connexion de tous les utilisateurs d'AsgardMenu sont bien membre de `g_consult`. Cf. II.3.A sur ce point.

V.2 — PARAMÉTRAGE D'ASGARDMENU

Dans QGIS, AsgardMenu peut être paramétré via le menu Extension > Asgard Menu > Asgard Menu param.



L'utilisateur doit choisir la connexion à utiliser parmi les connexions à un serveur PostgreSQL précédemment définies dans QGIS. La connexion sélectionnée doit pointer sur une base de données où l'extension ASGARD aura été installée, sans quoi AsgardMenu renvoie un message d'erreur.

Il faut ensuite cocher la case à gauche de « Menu principal » et/ou de « dock » pour que le menu apparaisse dans la barre des menus et/ou sous la forme d'un panneau. Cette dernière forme apporte une fonctionnalité qui permet de filtrer les tables par recherche dans leur commentaire.

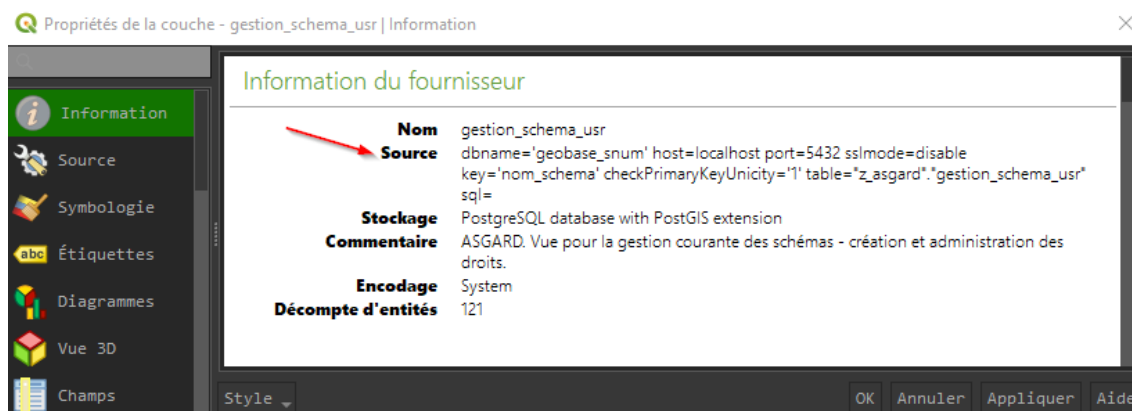
La validation peut se faire indifféremment avec le bouton Appliquer ou Enregistrer.

À ce stade, l'outil est mono-base. Si le service dispose de plusieurs bases équipées d'ASGARD, seule l'une d'entre elles pourra être utilisée pour générer le menu.

V.2.A. CONSIDÉRATIONS SUR LA DÉFINITION DE LA CONNEXION

Pour charger des tables et des vues via son menu, AsgardMenu s'appuie sur des identifiants de ressource unique (unique resource identifier ou URI en anglais) qui se composent de deux parties : une qui indique comment se connecter à la base – la chaîne de connexion – et une qui décrit la donnée.

Dans QGIS, ces URI se retrouvent dans l'onglet Information de la fenêtre Propriétés des couches PostgreSQL.



La partie chaîne de connexion, qui est celle qui importe ici, fait apparaître le nom de la base, l'adresse IP du serveur, le port, le mode SSL et, éventuellement, l'identifiant du rôle de connexion. Il est également possible de recourir à un « service » configuré dans un fichier `pg_service.conf` (cf. V.2.B pour ce cas particulier).

EXEMPLE. Chaîne de connexion standard pour l'utilisateur `jon.snow` sur un serveur distant.

Avec identifiant :

```
"dbname='geobase_snum' host=10.75.8.111 port=5432 user='jon.snow'
sslmode=require"
```

Sans identifiant :

```
"dbname='geobase_snum' host=10.75.8.111 port=5432 sslmode=require"
```

Idéalement, pour éviter à l'utilisateur de devoir saisir plusieurs fois son identifiant et son mot de passe, la configuration de la connexion à la base PostgreSQL dans QGIS devrait être parfaitement identique à celle qui apparaît dans les chaînes de connexion générées par ASGARD (qui permettent de charger les couches via le menu).

ASGARD saura déterminer automatiquement la plupart des informations nécessaires pour reconstituer les chaînes de connexion, mais certains aspects dépendent de stratégies locales. Dans ce cas, il sera du ressort de l'ADL de préciser la forme ou les valeurs attendues, via les champs de la table `asgard_parametre`, dans le schéma `z_asgard_admin`.

Cette table compte au plus un enregistrement, elle n'est éditée que par des membres de `g_admin` et les paramètres qui y sont renseignés valent pour tous les utilisateurs. Chaque champ contient la valeur d'un paramètre.

Lors de l'installation d'ASGARD, et tant qu'aucun paramètre n'a été spécifié manuellement, la table apparaît vide. Des valeurs par défaut sont définies pour tous les champs, et la manière la plus simple d'initialiser la table est encore d'appliquer ces valeurs par défaut :

```
INSERT INTO z_asgard_admin.asgard_parametre (id) VALUES (DEFAULT) ;
```

id	connex_param	sslmode	nom_service
[PK] integer	text[]	text	text
1	{host,port,dbname,sslmode}	require	[null]

L'initialisation selon la méthode précédente peut être faite sur n'importe quel champ et il est évidemment possible de saisir directement des valeurs personnalisées pour les paramètres. L'essentiel est que la première action sur la table soit une commande INSERT.

Par la suite, toutes les modifications de paramètres se feront par UPDATE.

On notera que le comportement d'ASGARD lorsque la table est vide est exactement le même que lorsque tous les champs prennent leur valeur par défaut. Autrement dit, aussi longtemps que les paramètres par défaut conviennent, il n'est pas nécessaire d'initialiser asgard_parametre.

Il est à tout moment possible de remettre un paramètre à sa valeur par défaut avec :

```
UPDATE z_asgard_admin.asgard_parametre
SET [nom du champ] = DEFAULT ;
```

Pour l'heure, la table de paramétrage compte trois champs (outre sa clé primaire) : `connex_param`, `sslmode` et `nom_service`.

En agissant sur la liste contenue dans le champ `connex_param`, l'administrateur peut choisir les mots clés qui apparaissent dans les chaînes de connexion.

Par défaut, ce champ prend la valeur suivante : `ARRAY['host', 'port', 'dbname', 'sslmode']`.

Cette liste indique que l'adresse du serveur, le port, le nom de la base et le mode SSL doivent être spécifiés par la chaîne de connexion. Sauf à définir un service (cf. V.2.B), ces mots clés correspondent à des informations indispensables et ne peuvent être supprimés de la liste. On pourra par contre ajouter et retirer librement le mot clé « user ». Lorsque ce dernier est présent, les identifiants des utilisateurs, soit les noms de leurs rôles de connexion, seront systématiquement ajoutés dans les chaînes de connexion d'AsgardMenu.

EXEMPLE. Pour faire apparaître le mot clé « user » dans les chaînes de connexion :

```
UPDATE z_asgard_admin.asgard_parametre
SET connex_param = array_append(connex_param, 'user') ;
```

Pour qu'il n'apparaisse plus :

```
UPDATE z_asgard_admin.asgard_parametre
SET connex_param = array_remove(connex_param, 'user') ;
```

Si user est présent dans la liste de connex_param, il sera attendu des utilisateurs qu'ils enregistrent leur identifiant (pas leur mot de passe !) lorsqu'ils configureront dans QGIS la connexion désignée comme « base » pour AsgardMenu. À défaut, il leur faudrait par la suite saisir systématiquement leur mot de passe à deux reprises : à l'ouverture de QGIS et lorsqu'ils chargent leur première couche via le menu.

Réciproquement, si user ne figure pas dans la liste de connex_param, les utilisateurs ne doivent pas enregistrer leur identifiant.

BONNE PRATIQUE. S'il est d'usage que les utilisateurs enregistrent dans QGIS leurs identifiants de connexion au serveur PostgreSQL, le paramètre connex_param de la table de paramétrage d'ASGARD est modifié par l'ajout du mot-clé user.

Par l'intermédiaire de la table de paramétrage d'ASGARD, l'administrateur peut également configurer le mode SSL⁴² pour les connexions **distantes**, en attribuant au champ sslmode l'une des valeurs suivantes : « disable », « allow », « prefer », « require », « verify-ca » et « verify-full ».

Par défaut, sslmode vaut « require ». Ce mode est compatible avec les serveurs PostgreSQL EOLE⁴³ et il paraît de judicieux de le préférer à disable, allow et prefer, qui ne garantissent pas (voire prohibent) l'établissement d'une connexion chiffrée.

42 Pour plus de précisions sur le support des connexions SSL et leur paramétrage côté client : <https://docs.postgresql.fr/10/libpq-ssl.html>.

43 Sur les serveurs EOLE, le paramètre ssl du fichier postgresql.conf est nativement configuré sur « on », permettant ainsi l'établissement de connexions chiffrées si le client (ici QGIS ou AsgardMenu) le demande.

BONNE PRATIQUE. Si, pour les connexions distantes au serveur, un mode SSL autre que *require/requiert* est utilisé, le champ *sslmode* de la table de paramétrage doit être modifié en conséquence.

Pour les connexions à un serveur **local**, « *disable* » sera retenu quelle que soit la valeur du champ *sslmode*, étant entendu qu'il n'est pas possible d'établir une connexion chiffrée en local.

En parallèle, dans la définition de la connexion QGIS, « *SSL mode* » doit être renseigné comme suit :

- ◆ pour une connexion distante, il est identique à la valeur de *sslmode* dans *asgard_parametre* (ou à sa traduction en français) ;
- ◆ pour une connexion locale, il vaut « *disable* ».

EXEMPLE. Configuration de la connexion QGIS de jon.snow à un serveur local avec `connex_param = ARRAY['host', 'port', 'dbname', 'sslmode', 'user']`. Il mémorise son identifiant (case Stocker cochée) et choisit « désactive » pour l'item « SSL mode ».

Dans ASGARD, il aura initialisé la table de paramétrage avec, par exemple :

```
INSERT INTO z_asgard_admin.asgard_parametre (id) VALUES (DEFAULT) ;
```

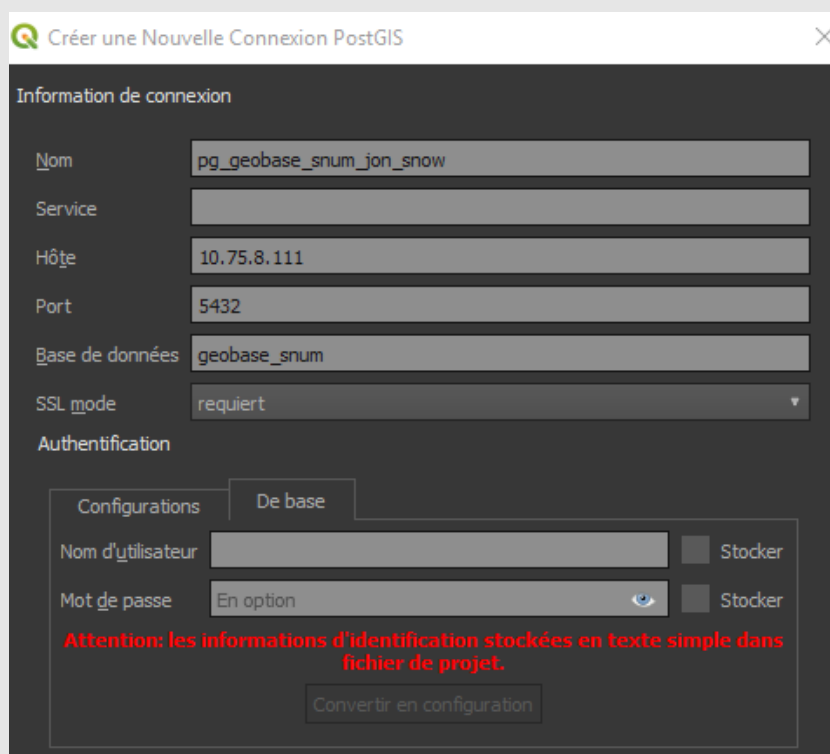
Il aura ensuite ajouté « user » à la liste des mots-clés devant figurer dans la chaîne de connexion :

```
UPDATE z_asgard_admin.asgard_parametre
SET connex_param = array_append(connex_param, 'user') ;
```

Ainsi, sa connexion dans QGIS sera cohérente avec les chaînes générées par ASGARD :

```
"dbname='geobase_snum' host=localhost port=5432 user='jon.snow'
sslmode=disable".
```

EXEMPLE. Configuration de la connexion de jon.snow à un serveur distant avec `connex_param = ARRAY['host', 'port', 'dbname', 'sslmode']` et `sslmode = 'require'` dans `asgard_parametre`. Il n'enregistre pas son identifiant et indique le mode SSL « requiert ».



Côté ASGARD, il n'a rien à faire, puisque ce paramétrage correspond au fonctionnement par défaut.

Les chaînes de connexions d'ASGARD sont alors :

```
"dbname='geobase_snum' host=10.75.8.111 port=5432 sslmode=require".
```

V.2.B. CONNEXION DÉFINIE PAR UN « SERVICE »

Dans QGIS, plutôt que de créer une connexion en spécifiant l'adresse du serveur, son port, le nom de la base, etc., il est possible d'avoir recours à un « service » qui fournira à QGIS tout ou partie des informations de connexion.

Concrètement, ces informations sont stockées dans un fichier `pg_service.conf`⁴⁴, dont le chemin est donné à QGIS par la variable `PGSERVICEFILE`⁴⁵.

Dans ce cas, les chaînes de connexion contiennent le mot-clé « service » (avec comme valeur associée le nom du service), ainsi que les éventuels mot-clés qui, au gré de l'administrateur, restent définis dans QGIS.

Dans la table de paramétrage d'ASGARD, on devra alors :

- ◆ indiquer le nom du service dans le champ `nom_service` ;
- ◆ retirer de `connex_param` les mot-clés dont les valeurs sont définies par le service et n'ont donc pas à apparaître dans les chaînes de connexion.

BONNE PRATIQUE. Si, dans QGIS, les connexions au serveur PostgreSQL sont définies à l'aide d'un « service », les paramètres `nom_service` et `connex_param` sont adaptés en conséquence.

Si « `sslmode` » ne figure pas dans la liste de `connex_param`, la valeur prise par le champ de même nom n'est pas considérée.

EXEMPLE. Supposons que l'adresse du serveur, l'hôte, la base et le mode SSL soient définis dans le fichier `pg_service.conf`. L'usage interne veut que les identifiants ne soient pas sauvegardés, ils ne sont donc renseignés ni dans le fichier de configuration de service, ni lors de la configuration de la connexion QGIS.

Dans le fichier de configuration de service, on trouve :

```
[service_geobase_snum]
host=10.75.8.111
port=5432
dbname=geobase_snum
sslmode=require
```

Les URI des couches QGIS, dont la chaîne de connexion ne contient que le mot-clé `service`, seront de la forme :

44 Pour plus de détails : <https://docs.postgresql.fr/10/libpq-pgservice.html>.

45 À ajouter le cas échéant comme variable personnalisée dans Préférences > Options, onglet Système.

```
"vector:postgres:journal_du_mur:service='service_geobase_snum'
table="w_snow"."journal_du_mur" sql=::::NoGeometry"
```

Dans ASGARD :

```
UPDATE z_asgard_admin.asgard_parametre
SET nom_service = 'service_geobase_snum',
    connex_param = NULL ;
```

Dans QGIS :

V.2.C. UTILISATION DES FICHIERS INI

Le paramétrage d'AsgardMenu est enregistré dans les fichiers de configuration de QGIS : QGIS3.ini (paramétrage local) ou qgis_global_settings.ini (paramétrage global)

```
[AsgardMenu]
```

```
database=pg_geobase_snum  
schema=z_asgard  
profile=consultation  
dock=false  
menubar=true
```

Les valeurs des paramètres schema et profile ne doivent pas être modifiées.

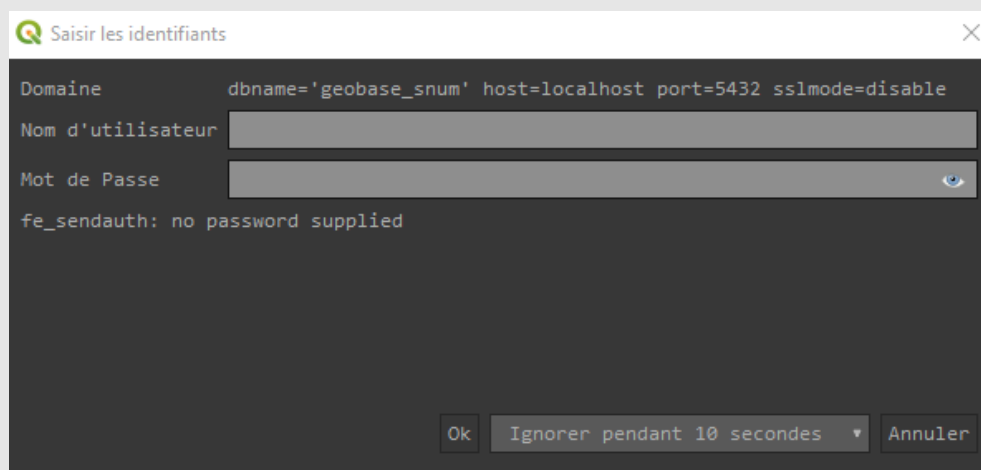
Les paramètres dock et menubar indiquent respectivement si le menu doit être affiché dans la barre des menus et sous forme de panneau. database donne le nom de la connexion à utiliser pour AsgardMenu, laquelle est d'ailleurs également spécifiée par l'un des fichiers de configuration.

En particulier, on pourrait imaginer qu'un fichier `qgis_global_settings.ini` commun à tous les agents du service définisse une connexion PostgreSQL et configure AsgardMenu avec celle-ci.

V.3 — UTILISATION COURANTE

Une fois paramétré, le plugin AsgardMenu se lance à l'ouverture de QGIS. Il demande alors le mot de passe de l'utilisateur pour la connexion pré-définie, ce qui lui permet d'accéder à la vue à partir de laquelle le menu est généré.

EXEMPLE. Connexion avec le paramétrage standard d'ASGARD (cf. V.2.A) :



Saisir les identifiants

Domaine dbname='geobase_snum' host=localhost port=5432 sslmode=disable

Nom d'utilisateur

Mot de Passe

fe_sendauth: no password supplied

Ok Ignorer pendant 10 secondes Annuler

Si les recommandations susmentionnées ont été suivies, l'utilisateur ne devrait plus avoir à saisir ses identifiants par la suite, sauf évidemment à ce qu'il aille chercher des données dans une autre base ou avec un autre rôle de connexion.

Il peut charger des couches dans QGIS en cliquant simplement sur les tables et vues présentées dans le menu. Comme dans l'explorateur de QGIS, le nom du champ de géométrie est précisé sous la forme `nom_table.nom_geometrie` s'il est différent de `geom` et les tables apparaissent autant de fois qu'elles ont de champs de géométrie.

Le menu est figé pour la durée de la session (jusqu'à la fermeture de QGIS) : si des objets sont créés, modifiés ou supprimés, l'utilisateur ne le verra pas. Il reste possible d'actualiser le menu en retournant dans `Extension > Asgard Menu > Asgard Menu param.` Après avoir saisi de nouveau les paramètres et cliqué sur `Appliquer`, le menu est mis à jour.

La même méthode peut être suivie pour générer de nouveau le menu avec un rôle de connexion différent : on choisira dans ce cas une « base » correspondant à une connexion paramétrée avec le nouveau rôle.

VI — INSTALLATION D'ASGARD

VI.1 — COMPATIBILITÉ

ASGARD est théoriquement compatible avec les versions 9.5 et supérieures de PostgreSQL. Il importe toutefois de noter que l'extension n'a à ce stade été testée de manière approfondie que sur des serveurs en versions 10 et 9.5.

En particulier, les nouveaux types d'objets rattachés à des schémas qui pourraient être introduits par des versions ultérieures ne sont pas pris en charge à ce stade : ASGARD n'appliquera automatiquement aucun droit sur ces objets, les fonctions de réinitialisation des droits ne les considéreront pas et la mise en cohérence automatique du propriétaire de l'objet avec le producteur du schéma ne sera pas assurée. Il reviendra donc à l'administrateur de données de réaliser ces opérations manuellement, le cas échéant.

Cf. aussi III.9 pour les objets non gérés des versions 10 et 9.5.

VI.2 — LIVRABLES

ASGARD est fourni sous la forme d'une extension pour PostgreSQL, soit deux fichiers :

- ♦ un script : `asgard--x.x.x.sql` (x.x.x étant le numéro de version) ;
- ♦ un fichier de définition des paramètres de l'extension : `asgard.control`.

VI.3 — INSTALLATION DE L'EXTENSION

Les fichiers `asgard-0.x.x.sql` et `asgard.control` doivent être copiés dans le répertoire d'installation des extensions de PostgreSQL.

EXEMPLE. Pour PostgreSQL 10 sous EOLE 2.7 :


```
/usr/share/postgresql/10/extension
```

Pour PostgreSQL 10 sous Windows :

```
C:\Program Files\PostgreSQL\10\share\extension
```

Pour activer ASGARD dans une base de données de PostgreSQL, un super-utilisateur doit lancer la commande SQL de création de l'extension :

```
CREATE EXTENSION asgard ;
```

Si, comme ci-avant, aucun numéro de version n'est précisé, c'est la version « par défaut » spécifiée par l'extension elle-même⁴⁶ qui sera installée.

Pour installer une version spécifique, il faut l'ajouter avec une clause VERSION.

EXEMPLE. Pour installer la version 0.6.1, antérieure à la version de référence :

```
CREATE EXTENSION asgard VERSION '0.6.1' ;
```

Il est possible d'obtenir la liste des versions disponibles pour l'installation avec la commande SQL suivante (ou en regardant les fichiers présents dans le répertoire des extensions) :

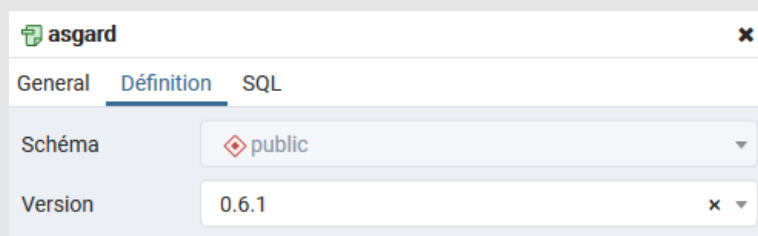
```
SELECT * FROM pg_available_extension_versions WHERE name = 'asgard' ;
```

La version effectivement installée est indiquée dans cette liste et peut par ailleurs être consultée via :

```
SELECT * FROM pg_available_extensions WHERE name = 'asgard' ;
```

⁴⁶ Définie dans le fichier asgard.control. Elle peut aussi être consultée avec la commande `SELECT * FROM pg_available_extensions WHERE name = 'asgard'.`

... ou encore dans les propriétés de l'extension :



L'extension PostGIS est requise pour ASGARD : la seconde ne pourra pas être installée en l'absence de la première.

EXEMPLE. Tentative d'installation sans l'extension PostGIS :

```
CREATE EXTENSION asgard ;
```

ERREUR: l'extension « postgis » requise n'est pas installée

HINT: Utilisez CREATE EXTENSION ... CASCADE pour installer également les extensions requises.

PostGIS s'installe de la même façon que PostgreSQL, avec une commande `CREATE EXTENSION postgis`. À partir de la version 9.6 de PostgreSQL, on pourra également installer les deux extensions en une seule commande :

```
CREATE EXTENSION asgard CASCADE ;
```

NOTICE: installation de l'extension requise « postgis »

```
CREATE EXTENSION
```

Concrètement, la dépendance d'ASGARD à PostGIS découle de l'utilisation par la vue `qgis_menubuilder_metadata` d'ASGARD de la vue `geometry_columns` de PostGIS pour générer les URI des tables.

VI.4 — INITIALISATION DE LA TABLE DE GESTION

À l'installation, la table de gestion d'ASGARD ne contient aucun schéma.

Si ASGARD est installé dans une base de données contenant déjà des schémas, il est possible d'intégrer les schémas souhaités dans la gestion d'ASGARD :

- ◆ en masse, par la fonction `asgard_initialisation_gestion_schema` (schéma `z_asgard_admin`), qui enregistre dans la table de gestion tous les schémas exis-

tants hors schémas système et hors une éventuelle liste d'exceptions fournie en argument de la fonction ;

- ◆ un par un, avec la fonction `asgard_initialise_schema` (schéma `z_asgard`).

Ces fonctions et leur usage sont évoquées plus en détails dans les parties IV.7 et IV.3 respectivement.

Dans la grande majorité des cas, `asgard_initialisation_gestion_schema` pourra être lancée sans risque sur l'ensemble des schémas et il est recommandé de le faire. Son seul effet concret, au-delà de l'enregistrement dans la table de gestion, est d'attribuer au propriétaire du schéma la propriété de tous les objets qui s'y trouvent, s'il n'en disposait pas déjà.

BONNE PRATIQUE. Après l'installation d'ASGARD, utiliser la fonction `asgard_initialisation_gestion_schema` pour initialiser sa table de gestion.

À noter que si certains schémas (ou des objets qu'ils contiennent) ont comme propriétaire un super-utilisateur, il sera nécessaire de lancer la fonction avec ledit super-utilisateur pour les référencer, ou de les exclure (temporairement) du référencement grâce à la liste d'exceptions.

À l'issue du référencement, on souhaitera souvent remettre en ordre les droits, si ce n'est tout remettre à plat pour repartir à zéro avec le système producteur/éditeur/lecteur d'ASGARD. Ce sujet est traité dans la partie III.12.

Par ailleurs, ASGARD permet d'importer tout ou partie de la nomenclature nationale dans la table de gestion, sous la forme de schémas pré-référencés qui pourront ensuite être créés. L'import se fait avec la fonction `asgard_import_nomenclature`, présentée dans la partie IV.9.

Si ASGARD est installé dans une base déjà peuplée et que le service entend normaliser le nom de ses schémas selon la nomenclature nationale, il est recommandé de procéder à cette opération avant d'initialiser la table de gestion avec les fonctions `asgard_import_nomenclature` et `asgard_initialisation_gestion_schema`. L'opération sera alors moins fastidieuse.

EXEMPLE. Intégration d'un schéma `transport_ferroviaire` qui, d'après la nomenclature, aurait vocation à s'appeler `c_tr_infra_ferroviaire`.

Si le nom du schéma est modifié en préalable, trois commandes simples amènent au résultat attendu :

```
-- import de la nomenclature :
```

```
SELECT z_asgard_admin.asgard_import_nomenclature() ;
-- changement de nom :
ALTER SCHEMA transport_ferroviaire
    RENAME TO c_tr_infra_ferroviaire ;
-- enregistrement dans la table de gestion :
SELECT z_asgard.asgard_initialise_schema('c_tr_infra_ferroviaire') ;
```

bloc	creation	nomenclature	niv1	niv2	nom_schema	lecteur
character varying (1)	boolean	boolean	character varying	character varying	character varying	character varying
c	true	true	transport_infrastructure	ferroviaire	c_tr_infra_ferroviaire	g_consult

Alors que si le schéma est intégré d'abord, il faut cinq commandes pour arriver au même résultat :

```
-- import de la nomenclature :
SELECT z_asgard_admin.asgard_import_nomenclature() ;
-- enregistrement du schéma dans la table de gestion :
SELECT z_asgard.asgard_initialise_schema('transport_ferroviaire') ;
```

bloc	creation	nomenclature	niv1	niv2	nom_schema	lecteur
character varying (1)	boolean	boolean	character varying	character varying	character varying	character varying
c	false	true	transport_infrastructure	ferroviaire	c_tr_infra_ferroviaire	g_consult
[null]	true	false	[null]	[null]	transport_ferroviaire	[null]

```
-- copie des données :
UPDATE z_asgard.gestion_schema_usr
    SET (niv1, niv1_abr, niv2, niv2_abr, nomenclature, lecteur) = (
        SELECT
            a.niv1,
            a.niv1_abr,
            a.niv2,
            a.niv2_abr,
            a.nomenclature,
            a.lecteur
        FROM z_asgard.gestion_schema_usr AS a
        WHERE a.nom_schema = 'c_tr_infra_ferroviaire'
    )
    WHERE nom_schema = 'transport_ferroviaire' ;
-- sortie de c_tr_infra_ferroviaire de la nomenclature
-- (nécessaire pour pouvoir le supprimer) :
UPDATE z_asgard.gestion_schema_usr
    SET nomenclature = False
    WHERE nom_schema = 'c_tr_infra_ferroviaire' ;
-- suppression :
DELETE FROM z_asgard.gestion_schema_usr
    WHERE nom_schema = 'c_tr_infra_ferroviaire' ;
-- changement de nom :
```

```
ALTER SCHEMA transport_ferroviaire  
  RENAME TO c_tr_infra_ferroviaire ;
```

VI.5 — MISE À JOUR DE L'EXTENSION

La mise à jour se fait selon le même procédé que l'installation :

- ◆ copie des fichiers de mise à jour dans le répertoire des extensions (cf. VI.3 pour plus de détails sur ce répertoire) ;
- ◆ activation de la mise à jour sur une base par un super-utilisateur, avec une commande SQL `ALTER EXTENSION asgard UPDATE`.

Les fichiers de mises à jour sont :

- ◆ un nouveau fichier `asgard.control`, qui remplacera le précédent ;
- ◆ un ou plusieurs fichiers `asgard--x.x.x--y.y.y.sql`, qui sont des scripts de mise à jour de la version `x.x.x` à la version `y.y.y` ;
- ◆ un fichier `asgard--y.y.y.sql`, qui contient l'intégralité du code de la nouvelle version. Celui-ci n'est pas utilisé lors de la mise à jour, mais pourra servir par la suite, notamment en cas de sauvegarde/restauration de la base, ou pour l'installation d'ASGARD sur une nouvelle base.

EXEMPLE. Pour installer la version 1.0.0 depuis une version antérieure :

```
ALTER EXTENSION asgard UPDATE TO '1.0.0' ;
```

Il est généralement plus simple de ne pas spécifier le numéro de la version cible, c'est dans ce cas celle qui est indiquée dans le fichier `asgard.control` qui sera utilisée.

```
ALTER EXTENSION asgard UPDATE ;
```

VI.6 — DÉSINSTALLATION DE L'EXTENSION

ASGARD peut être désinstallé avec la commande :

```
DROP EXTENSION asgard ;
```

Sauf à avoir été sauvegardées manuellement par ailleurs, toutes les informations contenues dans la table de gestion qui ne pourraient pas être déduites de la nomenclature nationale ou de l'état de la base (noms et producteurs des schémas créés) seront définitivement perdues.

Tous les objets créés par l'extension seront supprimés par la commande `DROP EXTENSION`, sauf les rôles (`g_admin`, `g_admin_ext`, `g_consult` et `consult.default`). Ceux-ci conservent également les privilèges attribués sur les objets qui n'appartenaient pas à l'extension.

VI.7 — DÉSACTIVER ASGARD ?

En principe, il n'existe qu'une situation dans laquelle ASGARD peut interférer négativement avec les actions d'un utilisateur : lorsqu'il a des raisons légitimes pour vouloir qu'un objet ait (et conserve) un propriétaire différent de celui du schéma qui le contient. Ce cas de figure est très hypothétique, d'autant qu'une bonne pratique dans ces circonstances serait de créer un nouveau schéma avec comme producteur le rôle en question.

À défaut, il est possible :

- ◆ de déréférencer a posteriori un schéma de la table de gestion avec la fonction `asgard_sortie_gestion_schema` (dans `z_asgard_admin`, utilisable uniquement par l'ADL, cf. IV.6). Ce schéma et son contenu ne seront alors plus pris en compte par aucun des mécanismes automatisés d'ASGARD ;
- ◆ de désactiver temporairement le déclencheur sur événement `asgard_on_create_schema`, afin que les schémas créés dans l'intervalle ne soient pas enregistrés dans la table de gestion. Seul le super-utilisateur peut procéder à cette désactivation et à la réactivation ultérieure.

Pour désactiver le déclencheur sur événement :

```
ALTER EVENT TRIGGER asgard_on_create_schema DISABLE ;
```

Pour le réactiver :

```
ALTER EVENT TRIGGER asgard_on_create_schema ENABLE ;
```

... ou, dans les propriétés de pgAdmin :

asgard_on_create_schema
✕

General
Définition
Sécurité
SQL

Activé ?
☒ Enable
☐ Disable
☐ Replica
☐ Always

Fonction déclencheur

z_asgard_admin.asgard_on_create_schema ✕ ▼

Événements
☐ DDL COMMAND START
☒ DDL COMMAND END
☐ SQL DROP

Quand

1 'CREATE SCHEMA'

VII — MISE EN ŒUVRE SUR UN EXEMPLE

VII.1 — DESCRIPTION DU CAS PRATIQUE

Le service SNUM souhaite mettre en place une base de données `geobase_snum` dédiée au domaine de l'habitat.

Outre des schémas de consultation de type `c_hab_vil_...`, cette base devra comporter des schémas référentiels contenant des données issues de la BDTopo (extrait pour les thèmes `e_bati`, `f_vegetation`, `h_administratif` et `i_zone_activite` à partir des données des formations en FOAD), ainsi qu'un schéma pour des données confidentielles.

Il existe au sein du SNUM une cellule géomatique dont les membres sont :

- ◆ Étienne Lousteau, chef de la cellule ;
- ◆ Michel Chrestien, adjoint.

Ils doivent avoir les droits d'administration sur la base.

Le service compte aussi un pôle étude et connaissances territoriales (PECT), composé des deux membres suivants :

- ◆ Eugénie Grandet, cheffe de pôle ;
- ◆ Félix Grandet, opérateur.

Les membres de ce pôle doivent pouvoir créer de nouveaux schémas. Dans un premier temps, ils auront notamment besoin d'un schéma `w_pect` partagé entre les deux membres du pôle. Eugénie Grandet voudrait également un schéma `w_eugenie_grandet` personnel pour faire quelques travaux.

Il existe par ailleurs de nombreux utilisateurs que l'on souhaite gérer avec un compte de connexion générique, pour leur donner accès aux schémas du domaine « habitat » et aux référentiels.

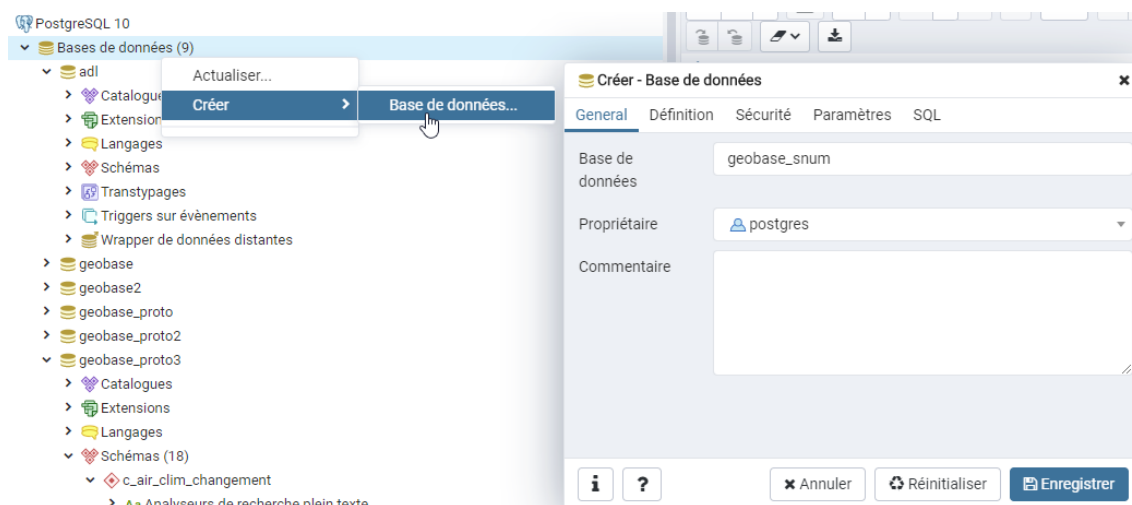
On se place dans la position d'Étienne Lousteau, chef de la cellule géomatique, qui peut se connecter au serveur avec un rôle super-utilisateur – postgres ou le rôle ad1 des serveurs EOLE.

VII.2 — PRÉPARATION DE LA BASE DE DONNÉES ET INSTALLATION D'ASGARD

Cette partie est une démonstration des manipulations à réaliser sur pgAdmin lors de l'installation d'ASGARD. Celles-ci ne seront toutefois possibles que si les fichiers d'ASGARD se trouvent déjà dans le répertoire des extensions du serveur. Cf. partie VI pour de plus amples explications sur la préparation de l'installation.

1. Se connecter avec un rôle super-utilisateur.
2. Créer la base de données geobase_snum.

Sous pgAdmin, par un clic droit sur « Bases de données » dans l'arborescence, puis Créer > Base de données :



... ou dans un éditeur SQL (ouvert sur n'importe quelle base pré-existante du serveur) :

```
CREATE DATABASE geobase_snum ;
```

3. Se connecter à la base geobase_snum et ouvrir l'éditeur de requêtes.

Par exemple par un clic droit sur le nom de la base dans l'arborescence, puis Éditeur de requêtes.

4. Créer l'extension PostGIS.

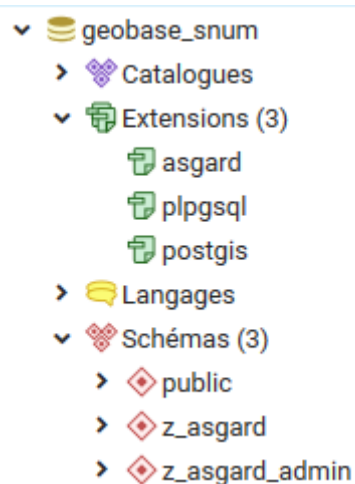
Comme expliqué au VI.3, la présence de PostGIS est en effet requise pour ASGARD.

```
CREATE EXTENSION postgis ;
```

5. Créer l'extension ASGARD.

```
CREATE EXTENSION asgard ;
```

On pourra vérifier dans l'arborescence de l'explorateur que l'extension asgard est maintenant listée parmi les extensions installées sur la base et que les schémas z_asgard et z_asgard_admin ont bien été créés :

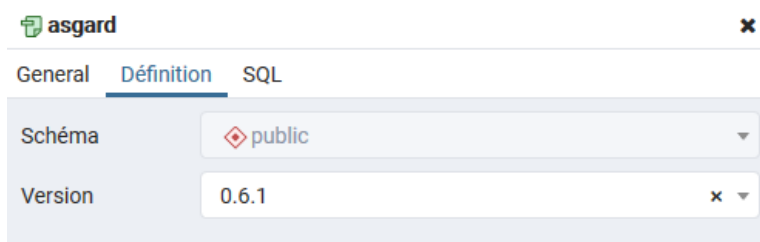


Les rôles g_admin, g_admin_ext, g_consult et consult.default doivent également apparaître dans la liste des rôles.

Pour connaître la version d'ASGARD qui vient d'être installée :

```
SELECT installed_version FROM pg_available_extensions
WHERE name = 'asgard' ;
```

... ou dans les propriétés de l'extension (interface pgAdmin) :



6. Faire de g_admin le propriétaire de geobase_snum.

Afin notamment de pouvoir ensuite octroyer à des rôles le droit de créer des schémas dans la base sans être contraint de passer par le rôle super-utilisateur, on fait de g_admin le propriétaire de geobase_snum.

```
ALTER DATABASE geobase_snum OWNER TO g_admin ;
```

Ce n'est en rien une obligation pour ASGARD, qui requiert seulement que g_admin puisse lui-même créer des schémas.

VII.3 — CRÉATION DES RÔLES ET INITIALISATION D'ASGARD

Pour la suite, on peut bien sûr utiliser les assistants de pgAdmin plutôt que les commandes SQL présentées.

7. Créer les rôles de connexion des membres de la cellule géomatique.

Ces rôles sont membres du rôle d'administration g_admin.

De plus, en tant qu'administrateurs du serveur, ils assureront tous deux la gestion des rôles. On leur octroie donc l'attribut CREATEROLE (cf. I.3.C pour de plus amples explications sur cette question).

```
CREATE ROLE "etienne.lousteau" CREATEROLE LOGIN PASSWORD
'etienne.lousteau';
GRANT g_admin TO "etienne.lousteau" ;
COMMENT ON ROLE "etienne.lousteau" IS 'ADMIN : Chef de la cellule
géomatique';

CREATE ROLE "michel.chrestien" CREATEROLE LOGIN PASSWORD
'michel.chrestien';
GRANT g_admin TO "michel.chrestien" ;
```

COMMENT ON ROLE "michel.chrestien" IS 'ADMIN : Adjoint de la cellule géomatique';

8. Se reconnecter avec le rôle de connexion etienne.lousteau.

D'une manière générale, il est préférable de restreindre l'usage du super-utilisateur aux cas où il est nécessaire.

Maintenant qu'Étienne Lousteau dispose d'un rôle de connexion individuel, il peut définir dans pgAdmin une nouvelle connexion pour ce rôle (clic droit sur « Serveurs » dans l'arborescence, puis Serveur > Créer serveur) et l'utiliser pour se reconnecter au serveur en tant que etienne.lousteau.

Créer - Serveur

General Connexion SSL Tunnel SSH Avancé

Nom: Lousteau

Groupe de serveurs: Servers

Arrière plan: ☐



Premier plan: ☐

Connecter maintenant ? ☒

Commentaires:

Nom de la connexion (au choix)

Annuler Réinitialiser Enregistrer

 **Créer - Serveur** 

General

Connexion

SSL

Tunnel SSH

Avancé

Nom d'hôte / Adresse

10.75.8.111

Port

5432

Base de données de maintenance

adl

Nom utilisateur

etienne.lousteau



Mot de passe


Enregistrer le mot de passe ?


☐


Rôle

Service

 Annuler

 Réinitialiser

 Enregistrer

Créer - Serveur

General

Connexion

SSL

Tunnel SSH

Avancé

Mode SSL

Require

Certificat client

Clé privée du client

Autorité de confiance du certificat

Liste des certificats révoqués

Compression SSL ?

No

i

?

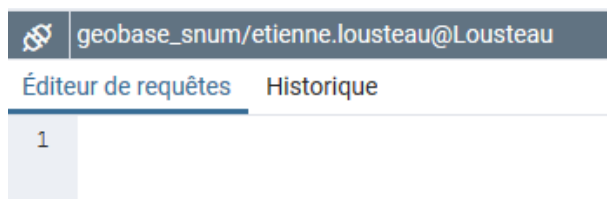
Annuler

Réinitialiser

Enregistrer

Les paramètres « Nom d'hôte/adresse », « Port », « Base de données de maintenance » et « Mode SSL » de la connexion sont à adapter selon le serveur. Les captures d'écran ci-avant montrent une configuration de connexion classique pour un serveur EOLE dont l'adresse IP serait 10.75.8.111.

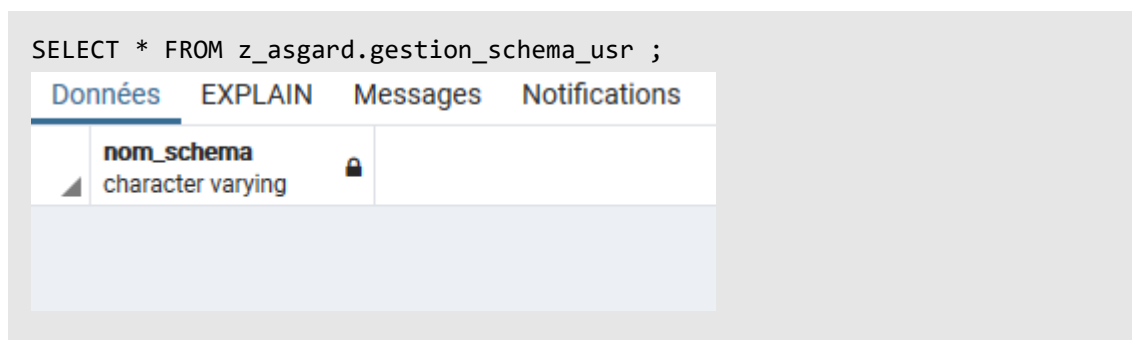
Si Étienne Lousteau ouvre cette nouvelle connexion dans l'explorateur, sélectionne la base `geobase_snum` et ouvre l'éditeur de requêtes, il voit maintenant⁴⁷ :



9. Référencer les schémas existants dans la table de gestion d'ASGARD

Avant de poursuivre sur les rôles, Étienne Lousteau procède à l'initialisation d'ASGARD.

Pour l'heure, s'il consulte le contenu de la table de gestion, il constatera qu'elle est vide :




Or, bien qu'elle vienne d'être créée, la base `geobase_snum` contient déjà deux schémas (en plus des schémas système non gérés par ASGARD) : `z_asgard` et `z_asgard_admin`, les schémas de l'extension.

Pour les référencer dans la table de gestion, Étienne Lousteau emploie la fonction `asgard_initialisation_gestion_schema` (cf. IV.7 pour le détail de son utilisation) :

```
SELECT z_asgard_admin.asgard_initialisation_gestion_schema() ;
```

⁴⁷ « Lousteau » correspond au nom donné au « serveur » lors de la définition de la connexion.


Données	EXPLAIN	Messages	Notifications
	asgard_initialisation_gestion_schema		
	text		
1	__ FIN INITIALISATION.		

Et s’il clique sur l’onglet « Messages », il verra le détail des opérations réalisées par ASGARD :

NOTICE: [table de gestion] Mise à jour du bloc pour le schéma z_asgard_admin (z).
 NOTICE: (schéma z_asgard_admin pré-existant)
 NOTICE: ... Schéma z_asgard_admin enregistré dans la table de gestion.
 NOTICE: [table de gestion] Mise à jour du bloc pour le schéma z_asgard (z).
 NOTICE: (schéma z_asgard pré-existant)
 NOTICE: ... Schéma z_asgard enregistré dans la table de gestion.

Si Étienne Lousteau visualise de nouveau le contenu de la table de gestion, il y trouvera maintenant les deux schémas z_asgard et z_asgard_admin.

```
SELECT bloc, nom_schema, producteur FROM z_asgard.gestion_schema_usr ;
```

Données	EXPLAIN	Messages	Notifications
	bloc	nom_schema	producteur
	character varying (1)	character varying	character varying
1	z	z_asgard_admin	g_admin
2	z	z_asgard	g_admin_ext

10. Importer le domaine habitat de la nomenclature

Étienne Lousteau sait qu’il lui faudra créer des schémas du domaine habitat de la nomenclature. Plutôt que de saisir leurs noms manuellement, il exécute la fonction dédiée d’ASGARD, `asgard_import_nomenclature`. Celle-ci ne crée pas immédiatement les schémas, mais prépare une création future en les pré-référençant dans la table de gestion d’ASGARD.

La fonction permet d’importer toute la nomenclature ou seulement certains domaines (cf. IV.9 pour sa description détaillée). Pour l’heure, Étienne Lousteau se limite au domaine habitat.


```
SELECT z_asgard_admin.asgard_import_nomenclature(
    ARRAY['habitat_politique_de_la_ville']
) ;
```

Données	EXPLAIN	Messages	Notifications
asgard_import_nomenclature text			
1	__ FIN IMPORT NOMENCLATURE.		

Dans l'onglet Messages, il voit la liste des schémas pré-enregistrés dans la table de gestion :

```
NOTICE: Le schéma c_hab_vil_access_propriete a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_besoin_logt a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_construction a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_habitat_indigne a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_occupation_logt a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_parc_loc_social a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_parc_logt a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_politique a été ajouté à la table de gestion.
NOTICE: Le schéma c_hab_vil_renovation a été ajouté à la table de gestion.
```

Et s'il regarde dans la table de gestion elle-même :

```
SELECT * FROM z_asgard.gestion_schema_usr ;
```

... il y retrouvera tous ces schémas, avec un champ creation valant False (schéma non créé), un producteur pré-identifié comme g_admin et un lecteur pré-identifié comme g_consult.

11. Créer le rôle de groupe du pôle étude, g_pect.

ASGARD étant désormais prêt à l'usage, Étienne Lousteau s'attaque maintenant à la préparation des rôles des agents du pôle étude.

Leur rôle de groupe dédié, g_pect, n'est pas un rôle de connexion et n'a pas l'attribut LOGIN⁴⁸.

48 On pourrait aussi écrire la requête suivante comme CREATE ROLE g_pect NOLOGIN. NOLOGIN étant la valeur par défaut, les deux formes sont équivalentes.

```
CREATE ROLE g_pect ;  
COMMENT ON ROLE g_pect IS 'Membres du Pôle Étude et Connaissances  
Territoriales';
```

Comme recommandé au II.3.B, Étienne Lousteau rend `g_admin` membre de ce rôle, qui sera bientôt amené à devenir producteur de schémas.

```
GRANT g_pect TO g_admin ;
```

À ce stade, `g_pect` n'a aucun droit sur la base, hormis ceux qui sont conférés à tous les utilisateurs via le pseudo-rôle public. Concrètement, il peut seulement accéder en lecture aux informations contenues dans les schémas `public`, `pg_catalog` et `information_schema`, et utiliser les fonctions et types qui y sont définis.

12. Créer les rôles de connexion individuels des membres du pôle étude.

Ces rôles sont membres de `g_pect`.

```
CREATE ROLE "eugenie.grandet" LOGIN PASSWORD 'eugenie.grandet' ;  
GRANT g_pect TO "eugenie.grandet" ;  
COMMENT ON ROLE "eugenie.grandet" IS 'Cheffe du PECT' ;  
  
CREATE ROLE "felix.grandet" LOGIN PASSWORD 'felix.grandet';  
GRANT g_pect TO "felix.grandet";  
COMMENT ON ROLE "felix.grandet" IS 'Adjoint du PECT';
```

13. Attribuer `g_consult` aux rôles de connexion.

Par réflexe, il est bon de déclarer dès maintenant l'ensemble des rôles de connexion créés comme membres de `g_consult` (cf. II.3.A).


On aurait pu lancer une commande `GRANT` pour chaque rôle – et cela aurait pu être fait au fur et à mesure de la création des rôles :

```
GRANT g_consult TO "felix.grandet" ;  
GRANT g_consult TO "eugenie.grandet" ;  
GRANT g_consult TO "michel.chrestien" ;  
GRANT g_consult TO "etienne.lousteau" ;
```

Mais, afin de gagner du temps, Étienne Lousteau choisit d'utiliser la fonction d'ASGARD `asgard_all_login_grant_role` pour attribuer `g_consult` à tous les rôles de connexion du serveur en une seule commande.

```
SELECT z_asgard_admin.asgard_all_login_grant_role('g_consult') ;
```

renvoie :

Données	EXPLAIN	Messages	Notifications
 asgard_all_login_grant_role integer			
1		4	

```
NOTICE: > GRANT g_consult TO "etienne.lousteau"
NOTICE: > GRANT g_consult TO "michel.chrestien"
NOTICE: > GRANT g_consult TO "eugenie.grandet"
NOTICE: > GRANT g_consult TO "felix.grandet"
```

À noter que si d'autres rôles de connexion (hors super-utilisateurs) avaient préalablement été créés sur le serveur, eux aussi deviendront membres de `g_consult`, et tel est bien le but recherché.

14. Créer un rôle de groupe habilité à créer des schémas.

Il est attendu que les membres de `g_pect` puissent créer des schémas. Comme on souhaite se réserver la possibilité d'accorder ultérieurement la même prérogative à d'autres groupes, on décide de créer un rôle de groupe `g_createschema`.

```
CREATE ROLE g_createschema ;
```

Deux stratégies sont alors possibles :

- ◆ soit on attribue directement `g_createschema` aux rôles de connexion des utilisateurs concernés, `eugenie.grandet` et `felix.grandet` ;
- ◆ soit on attribue `g_createschema` au rôle de groupe `g_pect`. Les rôles de connexion `eugenie.grandet` et `felix.grandet`, qui en sont membres, hériteront des droits de `g_pect` et donc de ceux de `g_createschema`.

La différence se fera sentir si un nouvel utilisateur est ajouté à `g_pect`. Dans le second cas, il pourra créer des schémas, pas dans le premier. La seconde option serait donc à retenir si et seulement s'il est certain que tous les membres actuels et futurs de `g_pect` devront pouvoir créer des schémas. On se place ici dans ce cas de figure.

```
GRANT g_createschema TO g_pect ;
```

Comme expliqué au III.1.B, en tant que rôle habilité à créer des schémas, `g_createschema` doit être le rôle éditeur du schéma `z_asgard` ou un membre de ce rôle. On retient ici la première option. Évidemment, il faudra aussi lui octroyer le privilège `CREATE` sur la base `geobase_snum`.

Mise à jour du rôle éditeur pour `z_asgard` :

```
UPDATE z_asgard.gestion_schema_usr
SET editeur = 'g_createschema'
WHERE nom_schema = 'z_asgard' ;
```

```
NOTICE: application des privilèges standards pour le rôle éditeur du schéma z_asgard :
NOTICE: > GRANT USAGE ON SCHEMA z_asgard TO g_createschema
NOTICE: > GRANT SELECT, UPDATE, DELETE, INSERT ON ALL TABLES IN SCHEMA z_asgard TO
g_createschema
NOTICE: > GRANT SELECT, USAGE ON ALL SEQUENCES IN SCHEMA z_asgard TO g_createschema
UPDATE 1
```

Droit `CREATE` sur la base pour `g_createschema` :

```
GRANT CREATE ON DATABASE geobase_snum TO g_createschema ;
```

15. Vérifier les privilèges de `felix.grandet`.

N'étant pas super-utilisateur, `etienne.lousteau` ne peut pas directement endosser le rôle `felix.grandet`.

```
SET ROLE "felix.grandet" ;
```

```
ERREUR: droit refusé pour configurer le rôle « felix.grandet »
```

S'il veut s'assurer que ce rôle peut désormais bel et bien créer des schémas, il lui faut demander à Félix Grandet de procéder aux tests, ou se reconnecter lui-même avec le super-utilisateur, dont l'une des caractéristiques est de pouvoir s'arroger tous les rôles (via `SET ROLE`), ou encore créer dans pgAdmin une nouvelle connexion avec l'identifiant et – tant qu'il le connaît encore – le mot de passe de `felix.grandet`.

Admettons qu'Étienne Lousteau choisisse une de ces méthodes.

Pour créer un schéma, `felix.grandet` doit, comme tous les utilisateurs, désigner un propriétaire qui soit un rôle de groupe existant dont il est membre. De plus, comme il n'est

pas habilité à créer des rôles (au contraire d'`etienne.lousteau`), il faut que `g_admin` soit également membre du groupe choisi⁴⁹. C'est le cas avec `g_pect`.

```
CREATE SCHEMA exemple AUTHORIZATION g_pect ;
```

NOTICE: [table de gestion] Le nom du schéma exemple ne respecte pas la nomenclature.

HINT: Si vous saisissez un préfixe dans le champ bloc, il sera automatiquement ajouté au nom du schéma.

NOTICE: ... Le schéma exemple a été enregistré dans la table de gestion.

CREATE SCHEMA

Les messages renvoyés par ASGARD soulignent que le nom du schéma n'est pas conforme à la nomenclature, néanmoins on peut vérifier que le schéma exemple est désormais présent dans la vue `gestion_schema_usr` et que son producteur est `g_pect`.

nom_schema character varying	creation boolean	producteur character varying	editeur character varying	lecteur character varying
exemple	true	g_pect	[null]	[null]

Mais si `felix.grandet` avait tenté la même requête avec le rôle `g_consult`, dont `g_admin` n'est pas membre, il aurait rencontré l'erreur suivante :

```
CREATE SCHEMA exemple_2 AUTHORIZATION g_consult ;
```

ERREUR: ECS0 > TA0 > TA6. Opération interdite. Permissions insuffisantes pour le rôle g_consult.

DETAIL: GRANT g_consult TO g_admin

HINT: Votre rôle doit être membre de g_consult avec admin option ou disposer de l'attribut CREATEROLE pour réaliser cette opération.

Supprimons physiquement le schéma exemple (toujours avec le rôle `felix.grandet`) :

```
DROP SCHEMA exemple ;
```

NOTICE: [table de gestion] Le nom du schema exemple ne respecte pas la nomenclature.

HINT: Si vous saisissez un préfixe dans le champ bloc, il sera automatiquement ajouté au nom du schéma.

NOTICE: ... La suppression du schéma exemple a été enregistrée dans la table de gestion (creation = False).

DROP SCHEMA

On peut vérifier que l'attribut `creation` vaut maintenant `False` pour le schéma exemple dans la vue `gestion_schema_usr`.

⁴⁹ Cf. II.3.B pour de plus amples précisions sur ce sujet.

nom_schema	creation	producteur	editeur	lecteur
character varying	boolean	character varying	character varying	character varying
exemple	false	g_pect	[null]	[null]

Supprimons définitivement l'enregistrement de la table de gestion :

```
DELETE FROM z_asgard.gestion_schema_usr WHERE nom_schema = 'exemple' ;
```

VII.4 —

MISE EN PLACE DES CONNEXIONS POSTGRESQL SOUS QGIS

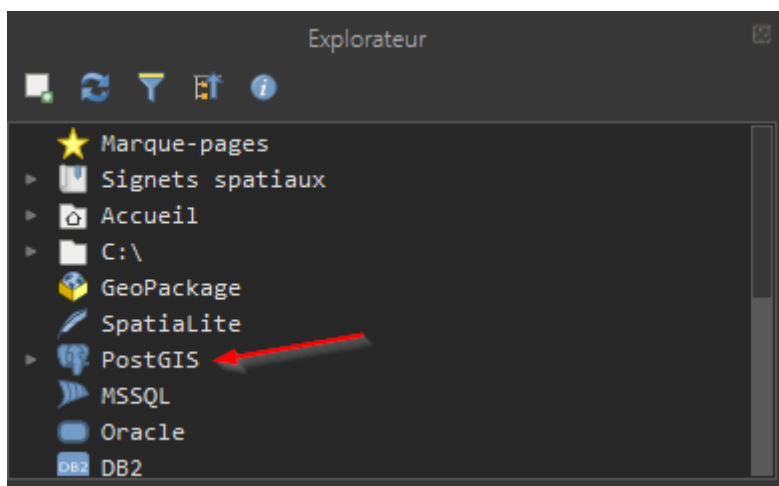
Il serait évidemment possible de poursuivre sous pgAdmin, mais ASGARD permet également de travailler sous QGIS et c'est cette option que nous allons à présent explorer.

Pour faire nos essais, nous allons créer différentes connexions PostgreSQL dans QGIS nous permettant d'adopter les différents rôles de connexion.

Les connexions PostgreSQL sont ajoutées sous QGIS en cliquant sur l'outil



... ou par un clic droit sur l'icône de PostGIS dans le panneau Explorateur, puis Nouvelle connexion :



16. Définir les connexions.

On configure des connexions rendant compte de la variété des profils de droits, soit une connexion pour chacun des rôles suivants :

- ◆ `consult.default` (utilisateur lambda de QGIS) ;
- ◆ `eugenie.grandet` (membre de `g_pect`, peut créer des schémas, disposera d'un schéma de travail individuel) ;
- ◆ `felix.grandet` (membre de `g_pect`, peut créer des schémas) ;
- ◆ `etienne.lousteau` (membre de `g_admin`).

L'option « Afficher les tables sans géométrie » doit être activée, sans quoi la vue utilisateur d'ASGARD ne sera pas visible.

Il est conseillé de donner à ces connexions des noms explicites, rappelant au moins le nom du rôle et le nom de la base qu'elles visent. Par exemple :

- ◆ `pg_geobase_snum_consult_default` ;
- ◆ `pg_geobase_snum_etienne_lousteau` ;
- ◆ `pg_geobase_snum_eugenie_grandet` ;
- ◆ `pg_geobase_snum_felix_grandet`.

Enfin, pour passer plus aisément d'un rôle à l'autre, on choisira ici de mémoriser les noms des rôles dans le paramétrage.

Comme expliqué au point V.2.A, il convient de paramétrer en conséquence les chaînes de connexion d'ASGARD. Pour cela, on revient brièvement dans pgAdmin avec le rôle `etienne.lousteau` afin d'exécuter les deux commandes suivantes.

Initialisation de la table de paramétrage :

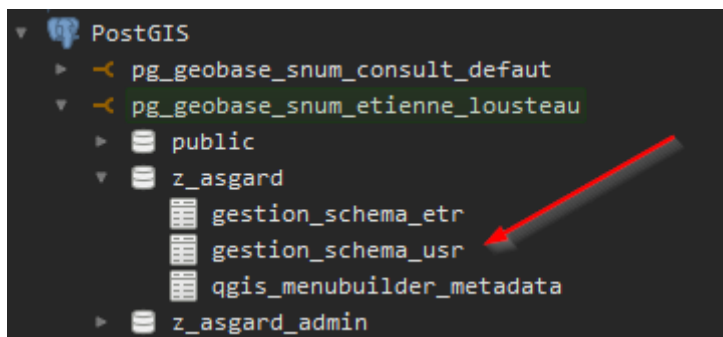
```
INSERT INTO z_asgard_admin.asgard_parametre (id) VALUES (DEFAULT) ;
```

Ajout de l'identifiant à la liste des mots-clés à faire figurer dans les chaînes de connexion :

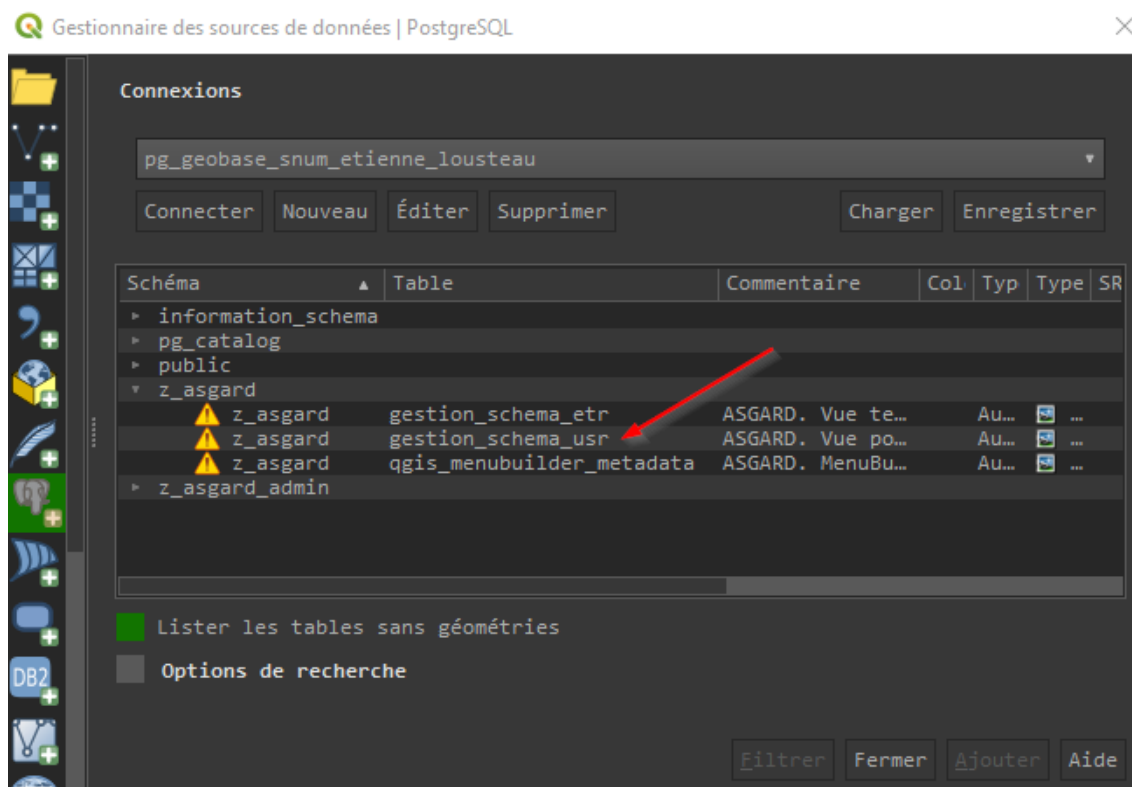
```
UPDATE z_asgard_admin.asgard_parametre  
SET connex_param = array_append(connex_param, 'user') ;
```

17. Charger la vue `gestion_schema_usr` avec `etienne.lousteau`.

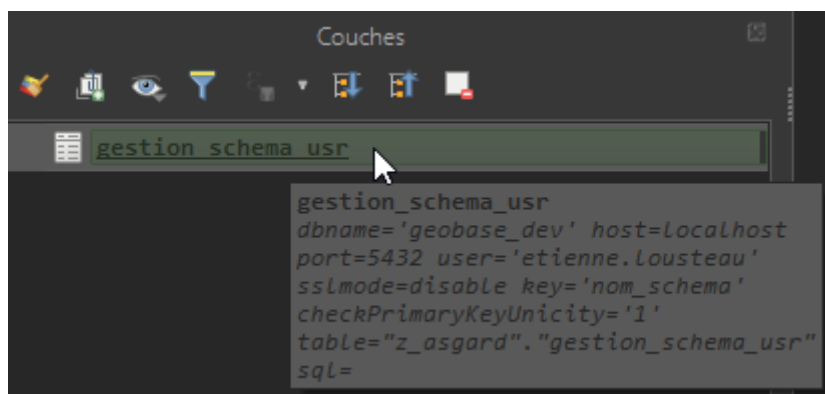
On pourra passer par le panneau Explorateur en choisissant bien la connexion `pg_geobase_snum_etienne_lousteau`.



... ou par le menu Couche > Ajouter une couche > Ajouter des couches PostGIS, en cliquant sur le bouton Ajouter :



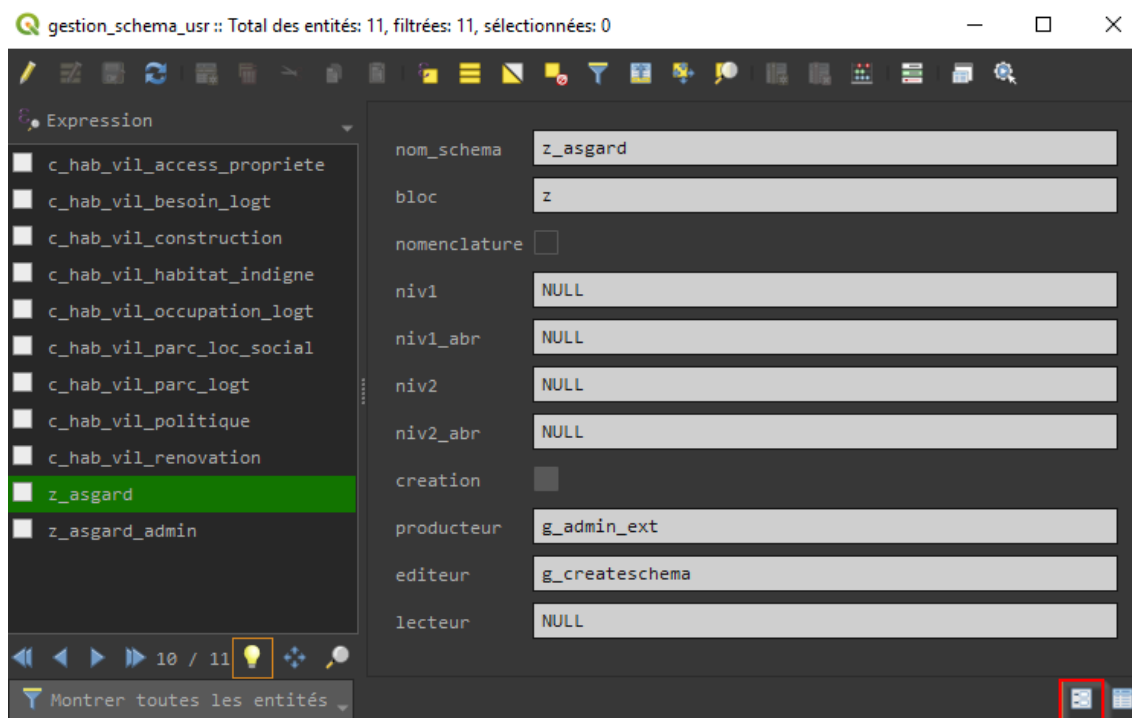
L'info-bulle qui apparaît après quelques secondes d'arrêt sur le nom de la couche dans le panneau Couches permet de vérifier la chaîne de connexion et en particulier, lorsque l'identifiant a été mémorisé dans la définition de la connexion, le paramètre user (ici etienne.lousteau).



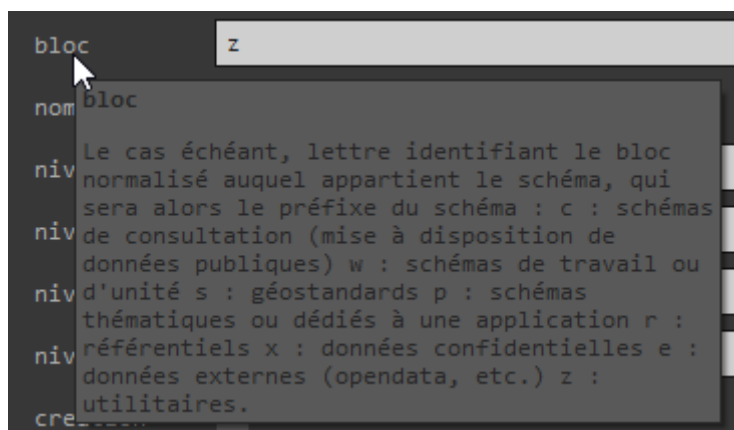
18. Ouvrir la table d'attributs de la couche.

etienne.lousteau étant membre de g_admin, la vue gestion_schema_usr affiche tous les schémas référencés dans la table de gestion lors de l'initialisation d'ASGARD.

Il est conseillé de passer en vue formulaire (bouton en bas à droite).

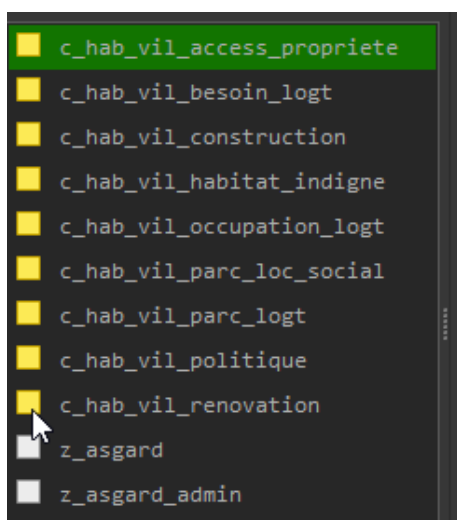


On notera que le descriptif d'un champ s'affiche lorsqu'on fige le curseur sur son nom.




19. Sélectionner tous les schémas du domaine habitat.

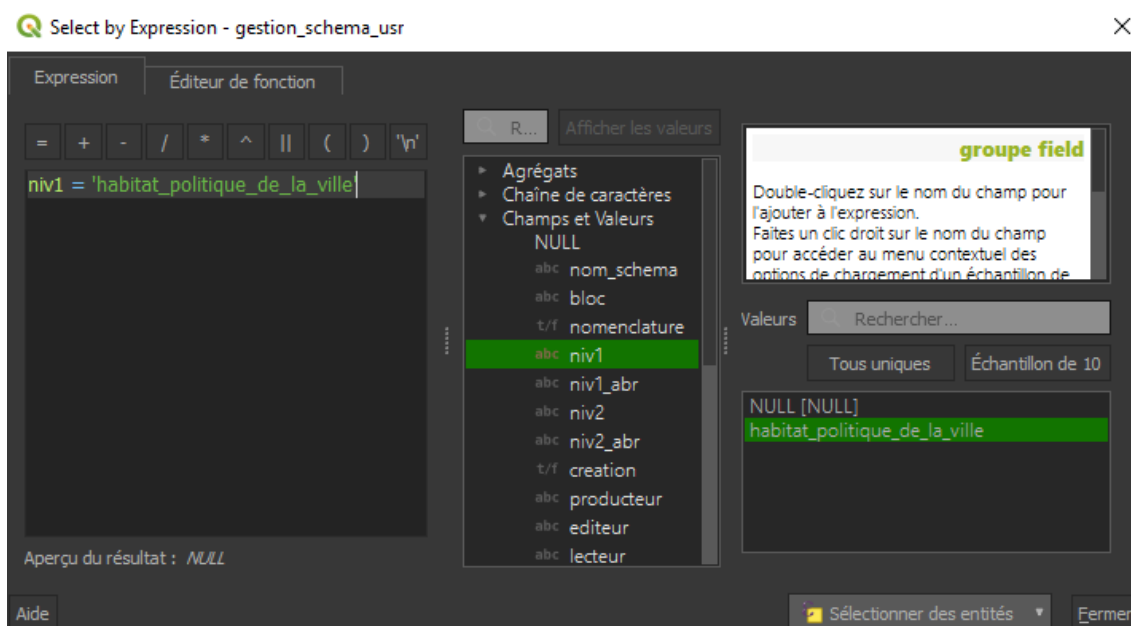
Ici, on pourra maintenir la touche Ctrl enfoncée tout en cliquant sur les carrés qui précèdent les noms de chacun des schémas concernés dans la liste à gauche de la fenêtre :



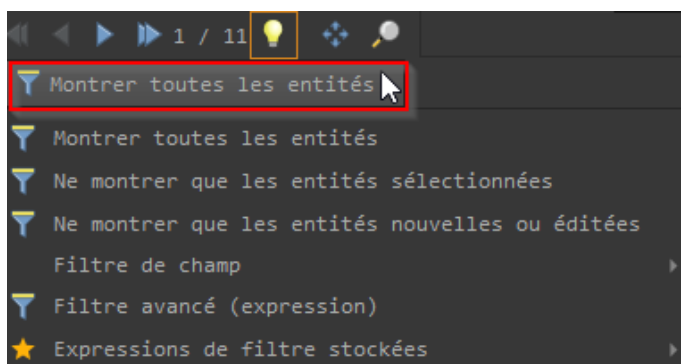
L'outil Sélection par expression permet également de sélectionner rapidement tous les schémas selon la valeur prise par le champ niv1.

 gestion_schema_usr :: Total des entités: 11, filtrées: 11, sélectionnées: 0



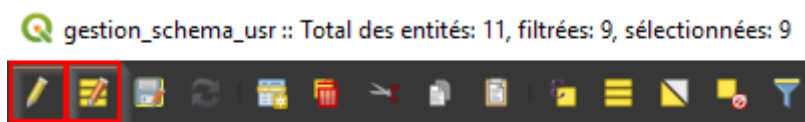


On peut ensuite choisir « Ne montrer que les entités sélectionnées » dans les options de filtrage (en bas à gauche de la fenêtre).

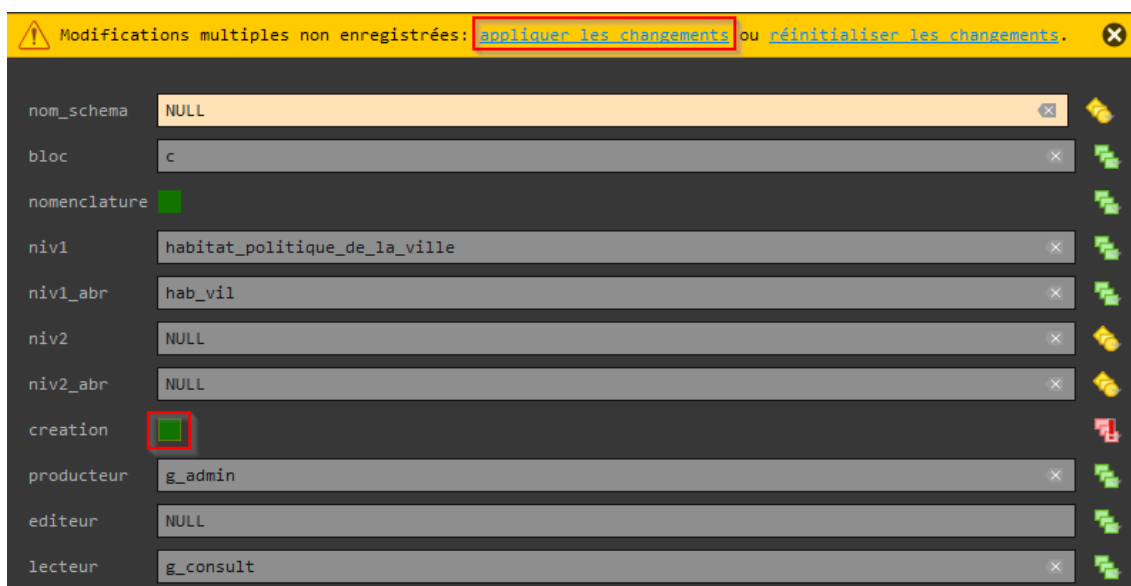


20. Activer la création des schémas du bloc habitat.

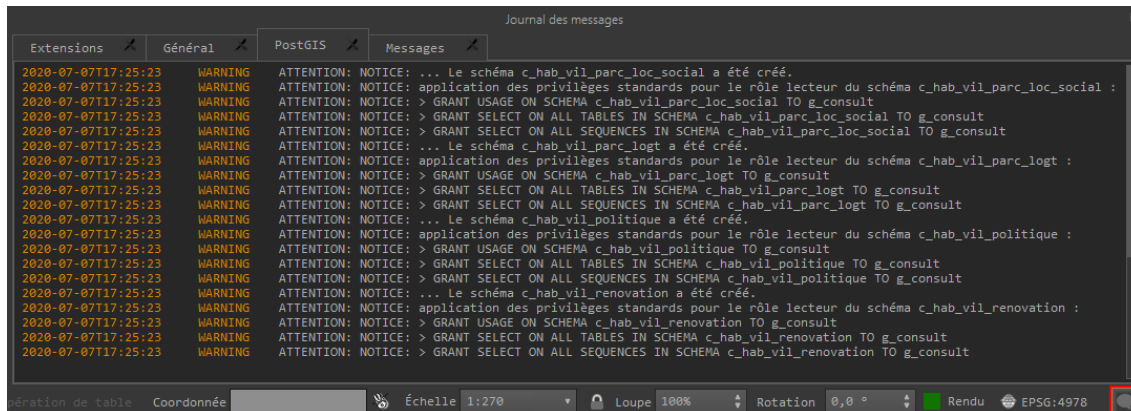
Pour ce faire, on passe en mode édition, puis on bascule en mode d'édition multiple :



Pour créer les 9 schémas sélectionnés, il faut alors cocher la case creation, puis cliquer sur « appliquer les changements » dans le bandeau jaune et quitter le mode édition.



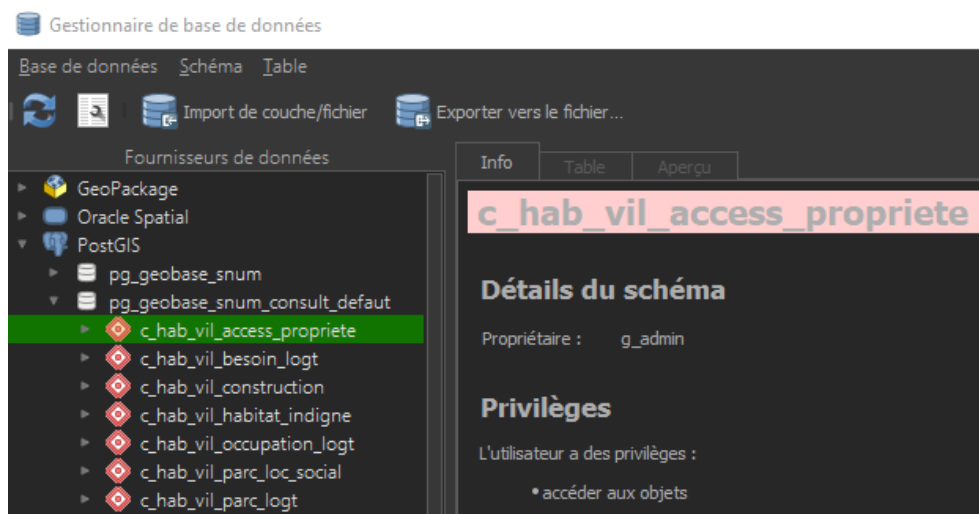
Les messages générés par ASGARD pour rendre compte à l'utilisateur des actions réalisées peuvent être consultés dans le journal de messages de QGIS (bouton tout en bas à droite de la fenêtre principale), onglet PostGIS.



En l'occurrence, ils informent de la création des schémas et de l'application des droits du rôle renseigné comme lecteur (g_consult).

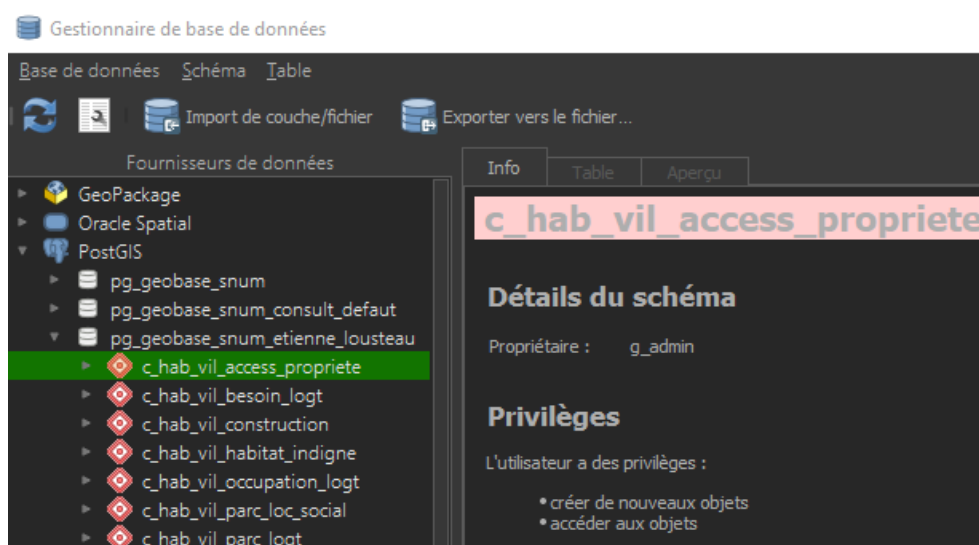
21. Comparer les privilèges de chacun via DBManager.

Lorsqu'on active la connexion geobase_snum_consult_default dans DBManager, on peut constater que consult.default a accès aux objets du schéma c_hab_vil_access_propriete. Il en va de même pour eugenie.grandet.



En effet, ils sont membres de `g_consult`, lecteur de ces schémas.

`etienne.lousteau` dispose en plus du privilège « créer de nouveaux objets », puisqu'il est membre du rôle `g_admin`, producteur du schéma.



VII.5 — FINALISATION DE LA CRÉATION DES SCHÉMAS ET IMPORT D'OBJETS

Étienne Lousteau doit maintenant créer des schémas pour les données référentielles issues de la BDTopo. Personne ne doit pouvoir changer les objets de ces schémas hormis les administrateurs. Il choisit donc `g_admin` comme groupe producteur, ne désigne pas d'éditeur et indique `g_consult` comme lecteur (tout le monde).

22. Créer les schémas référentiels.

Concrètement, `etienne.lousteau` ajoute un nouvel enregistrement dans la vue `gestion_schema_usr` pour chacun des quatre schémas qu'il souhaite créer (outil « Ajouter une entité »).

Qgis gestion_schema_usr :: Total des entités: 11, filtrées: 11, sélectionnées: 0



Pour le bâti :

nom_schema	r_bdtopo_bati	✕	✓
bloc	r	✕	
nomenclature			
niv1	BDTopo	✕	
niv1_abr	NULL		
niv2	Bâti	✕	
niv2_abr	NULL		
creation	<input checked="" type="checkbox"/>		
producteur	g_admin	✕	
editeur	NULL		
lecteur	g_consult	✕	

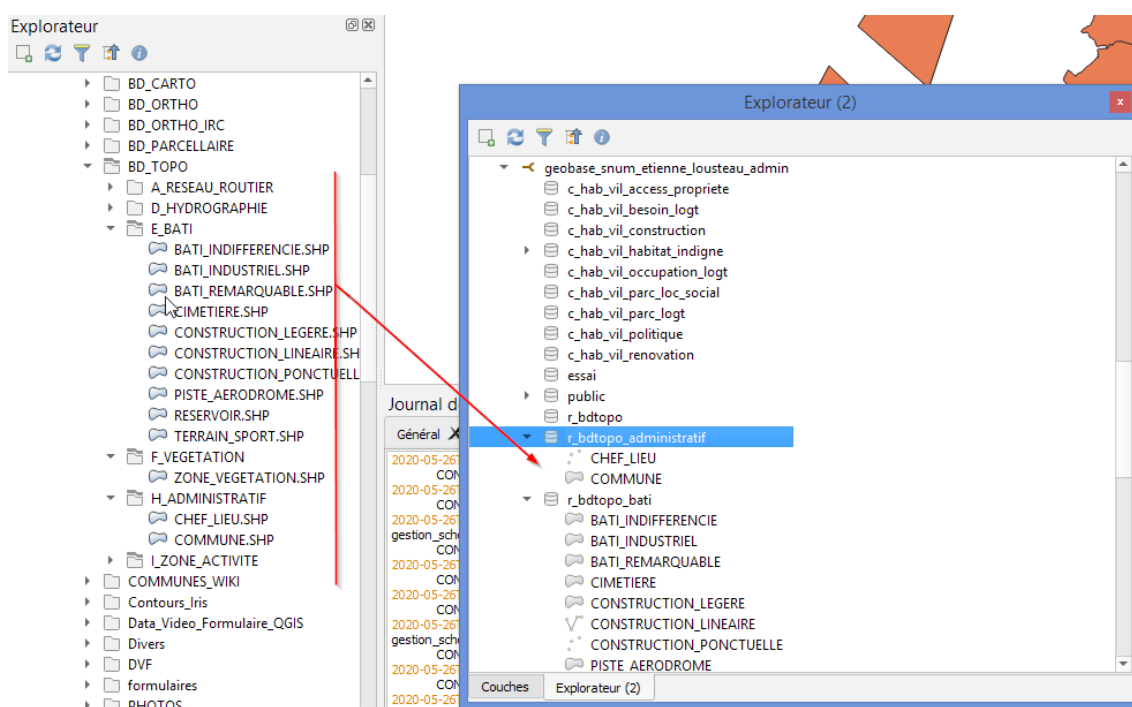
Puis il fait de même avec les thèmes `vegetation`, `administratif`, `zone_activite`.

On peut vérifier dans l'onglet PostGIS du journal des messages de QGIS que les créations se sont bien déroulées.

23. Importer les données.

Étienne Lousteau peut désormais importer des données dans les schémas à partir des données de la formation FOAD (data_foad).

Pour cela, on peut par exemple utiliser le deuxième explorateur de QGIS (peut être activé après un clic droit dans un espace libre de la barre d'outil de la fenêtre principale) pour faire des glisser/lâcher.



24. Créer un schéma pour les données confidentielles.

Étienne Lousteau décide de créer un groupe `g_confidentiel`, dont les membres seront les seuls à avoir accès aux objets d'un schéma nommé `x_confidentiel` (en accord avec la nomenclature).

Les membres de `g_confidentiel` devront pouvoir créer des objets dans le schéma, ce qui implique que ce rôle en soit le producteur.

On notera que `g_confidentiel` n'existe pas encore. Toutefois `g_admin` dispose du droit `CREATEROLE` et `etienne.lousteau`, qui en est membre, pourra donc créer directement des rôles au travers de la vue `gestion_schema_usr`.

nom_schema	x_confidentiel	x	✓
bloc	x	x	
nomenclature			
niv1	ACCES RESTREINT	x	
niv1_abr	NULL		
niv2	NULL		
niv2_abr	NULL		
creation			
producteur	g_confidentiel	x	
editeur	NULL		
lecteur	NULL		

Après validation et enregistrement, ASGARD renvoie :

```
NOTICE: ... Le rôle de groupe g_confidentiel a été créé.
NOTICE: ... Permission accordée à g_admin sur le rôle g_confidentiel.
NOTICE: ... Le schéma x_confidentiel a été créé.
INSERT 0 1
```

Pour l'instant personne n'est membre de `g_confidentiel` hormis `g_admin`, car ASGARD rend automatiquement `g_admin` membre de tous les groupes producteurs.

Si un utilisateur qui ne serait pas membre de `g_admin` doit travailler dans le schéma `g_confidentiel`, il suffira de le rendre membre de `g_confidentiel` (sous pgAdmin ou par une commande SQL dans DBManager sous QGIS).

```
GRANT g_confidentiel TO "esther.gobseck" ;
```

25. Créer les schémas du pôle étude avec eugenie.grandet.

Eugénie Grandet doit maintenant créer le schéma `w_pect` et son schéma personnel `w_eugenie_grandet`.

On veillera à bien ré-ouvrir la vue `gestion_schema_usr` avec la connexion correspondant au rôle `eugenie.grandet`.

Le schéma `w_pect` ne pose pas de difficulté particulière. Son producteur sera `g_pect`, afin que les agents du pôle puissent y créer des objets. Comme il s'agit d'un schéma de travail, il ne paraît pas utile de lui définir un lecteur ou un éditeur.

Pour le schéma `w_eugenie_grandet`, se pose la question du rôle producteur. En effet, ASGARD n'autorise pas la désignation d'un rôle de connexion comme propriétaire d'un schéma.

Ainsi, une tentative de création du schéma avec `eugenie.grandet` comme rôle producteur échoue, avec le message suivant (dans le journal de messages de QGIS) :

```
ERREUR: ECS0 > TA0 > TA3. Opération interdite (schéma w_eugenie_grandet). Le producteur/propriétaire du schéma ne doit pas être un rôle de connexion.
```

Il faudrait donc qu'Eugénie Grandet puisse disposer d'un nouveau rôle de groupe, par exemple `g_eugenie_grandet`, dont elle sera la seule membre (avec `g_admin`) et qui sera producteur du schéma `w_eugenie_grandet`.

Toutefois, elle n'est pas habilitée à créer des rôles.

Les administrateurs pourraient décider de lui octroyer ce droit en conférant l'attribut `CREATEROLE` à son rôle de connexion.

Cependant accorder un tel privilège sur le serveur simplement pour permettre à Eugénie Grandet de créer un unique rôle à un instant donné est une mauvaise pratique (cf. partie I.3.C pour de plus amples explications). Il est préférable que l'un des administrateurs se charge de créer le rôle `g_eugenie_grandet` pour elle (et valide parallèlement cette stratégie consistant à utiliser des rôles de groupes individuels).

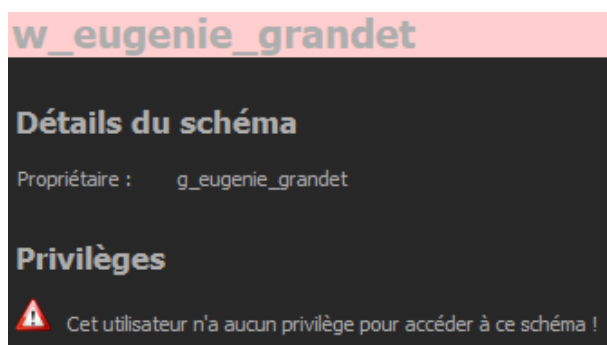
```
CREATE ROLE g_eugenie_grandet ;  
COMMENT ON ROLE g_eugenie_grandet IS 'Réserve à Eugénie Grandet, PECT' ;  
GRANT g_eugenie_grandet TO g_admin ;  
GRANT g_eugenie_grandet TO "eugenie.grandet" ;
```

Dès lors, Eugénie Grandet pourra créer son schéma dans QGIS en passant par `gestion_schema_usr`.

On peut vérifier les droits sous DBManager :



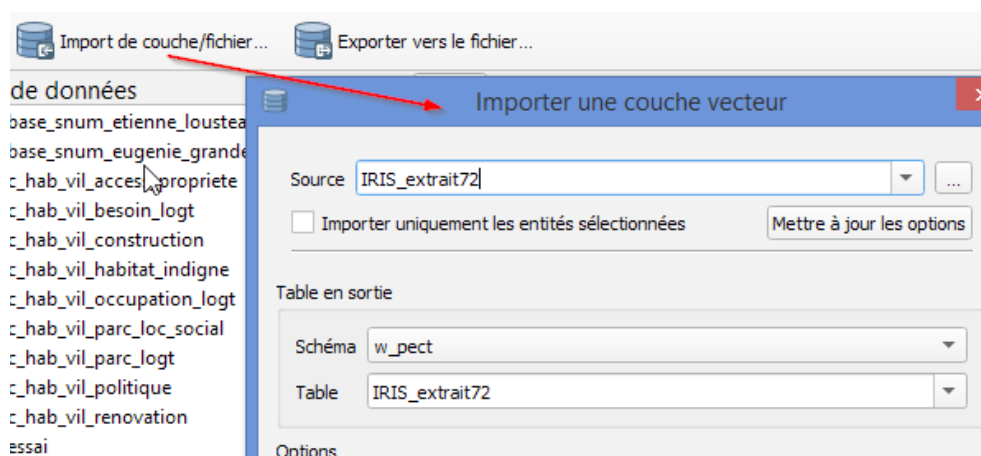
Alors que pour son collègue Félix Grandet, on a :



26. Importer une table dans le schéma w_pect.

Eugénie Grandet souhaite importer la table IRIS_extrait72 dans le schéma w_pect.

Elle peut le faire par exemple avec DBManager, en ayant préalablement chargé la table sous QGIS (data_foad/Contours_Iris/IRIS_extrait72.shp).



27. Mettre la table IRIS_extrait72 en consultation.

Après travail (ici virtuel !) sur cette couche, Eugénie Grandet souhaite publier la couche sous `c_hab_vil_occupation_logt`. Elle n'a pas les droits pour le faire. Elle doit donc demander à Étienne Lousteau qui est administrateur (membre de `g_admin`).

Ce dernier considère que les droits sur la table déplacée devront correspondre aux privilèges standards pour le schéma `c_hab_vil_occupation_logt`. Il procède donc au transfert avec la variante n°2 de la fonction `asgard_deplace_obj`.

```
SELECT z_asgard.asgard_deplace_obj('w_pect', 'IRIS_extrait72', 'table',
  'c_hab_vil_occupation_logt', 2) ;
```

Données	EXPLAIN	Messages	Notifications
asgard_deplace_obj text			
1	__ DEPLACEMENT REUSSI.		

NOTICE: attribution de la propriété de `c_hab_vil_occupation_logt.IRIS_extrait72` au rôle producteur du schéma :

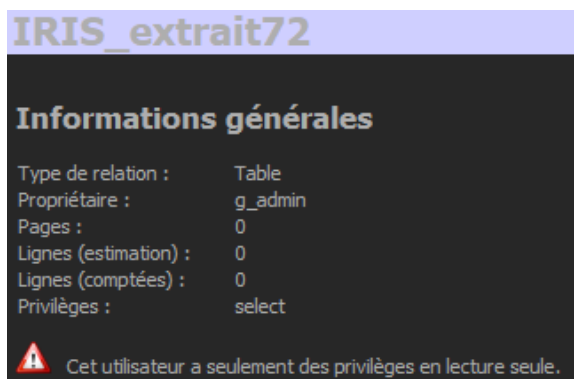
NOTICE: > ALTER table `c_hab_vil_occupation_logt`."IRIS_extrait72" OWNER TO `g_admin`

NOTICE: ... Objet déplacé dans le schéma `c_hab_vil_occupation_logt`.

NOTICE: application des privilèges standards pour le rôle lecteur du schéma :

NOTICE: > GRANT SELECT ON TABLE `c_hab_vil_occupation_logt`."IRIS_extrait72" TO `g_consult`

Il peut être vérifié qu'Eugénie Grandet ne dispose plus sur cette table que d'un accès en lecture, hérité de g_consult.



VII.6 — MENU POUR QGIS

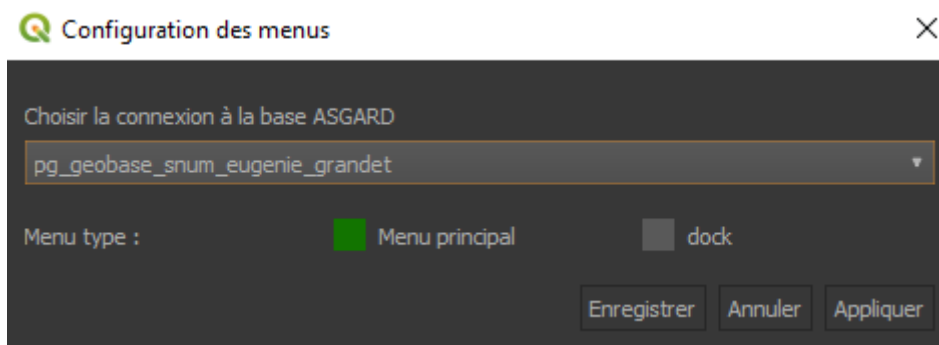
28. Si ce n'est pas déjà fait, télécharger le plugin AsgardMenu.

29. Configurer le menu.

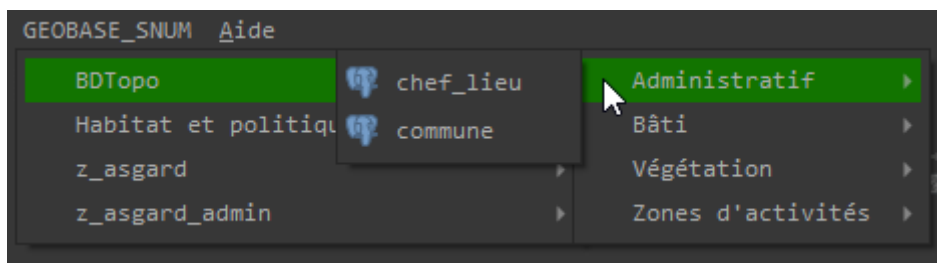
Le paramétrage doit être réalisé dans Extension > Asgard Menu > Asgard Menu param.

Le nom de la connexion à utiliser dépendra de l'utilisateur.

Prenons pour exemple Eugénie Grandet :



Le menu GEOBASE_SNUM est désormais disponible :



Parmi les schémas précédemment créés, Eugénie Grandet ne voit dans son menu que ceux :

- ◆ sur lesquels elle dispose a minima de droits de lecture (donc pas x_confidentiel, par exemple) ;
- ◆ qui contiennent effectivement des objets. Ainsi, les schémas vides w_pect et w_eugenie_grandet n'apparaissent pas.

ANNEXE A — FONCTIONS SUPPLÉMENTAIRES

Les fonctions présentées dans cette annexe interviennent directement dans les mécanismes automatisés d'ASGARD. Au contraire de celles du IV, elles n'ont pas été développées avec pour objectif de répondre à des cas d'usage ou de compléter les fonctionnalités d'ASGARD. Pour autant, elles peuvent être pertinentes dans certains contextes d'administration et rien ne s'oppose à leur utilisation « manuelle ».

ASGARD_ADMIN_PROPRIETAIRE

```
z_asgard.asgard_admin_proprietaire(n_schema text, n_owner text [,  
b_setschema boolean DEFAULT true ] )
```

La fonction `asgard_admin_proprietaire` permet d'attribuer un schéma et tous les objets qu'il contient à un [nouveau] propriétaire.

SCHÉMA : `z_asgard`.

ARGUMENTS :

- (1) ***n_schema*** est une chaîne de caractères correspondant au nom du schéma à considérer ;
- (2) ***n_owner*** est une chaîne de caractères correspondant au nom du rôle qui sera rendu propriétaire des objets ;
- (3) [optionnel] ***b_setschema*** est un paramètre booléen qui indique si la fonction doit changer le propriétaire du schéma et son contenu (True) ou seulement des objets qu'il contient (False). True par défaut.

RÉSULTAT : lorsque l'exécution est terminée, la commande renvoie le nombre d'objets traités. Les commandes lancées sont notifiées au fur et à mesure et peuvent être consultées dans l'onglet « Messages » de l'éditeur de requêtes de pgAdmin.

En elle-même, cette fonction n'agit que sur les propriétaires : elle liste l'ensemble des objets dont le propriétaire doit être réassigné et lance les commandes ALTER [OBJET] ... OWNER TO pour ce faire. Néanmoins, dans le contexte d'ASGARD, elle aura indirectement



un effet supplémentaire : le producteur du schéma sera mis à jour dans la table de gestion.

Pour l'utilisateur, sauf à l'appliquer à un schéma qui ne serait pas référencé dans ASGARD, elle produit ainsi exactement le même résultat qu'une commande `ALTER SCHEMA ... OWNER TO`, ce qui présente peu d'intérêt. Elle est par contre essentielle pour ASGARD, car c'est par elle que passent toutes les opérations de réassignation de propriétaire lorsque le producteur d'un schéma est modifié⁵⁰.

EXEMPLE.

```
SELECT z_asgard.asgard_admin_proprietaire('c_bibliotheque', 'g_snum') ;
```

produit le résultat

Données	EXPLAIN	Messages	Notifications
 asgard_admin_proprietaire integer			
1		3	

avec dans l'onglet Messages

```
NOTICE: attribution de la propriété des objets au rôle producteur du schéma c_bibliotheque :
NOTICE: > ALTER TABLE c_bibliotheque.journal_du_mur OWNER TO g_snum
NOTICE: > ALTER SEQUENCE c_bibliotheque.journal_du_mur_id_seq OWNER TO g_snum
NOTICE: ... Le propriétaire du schéma c_bibliotheque a été mis à jour dans la table de gestion.
NOTICE: > ALTER SCHEMA c_bibliotheque OWNER TO g_snum
NOTICE: > ALTER TABLE c_bibliotheque.journal_du_mur OWNER TO g_snum
NOTICE: > ALTER SEQUENCE c_bibliotheque.journal_du_mur_id_seq OWNER TO g_snum
```

ASGARD_GRANT_TO_REVOKE

```
z_asgard.asgard_grant_to_revoke(c_grant text)
```

⁵⁰ Ce qui, dans l'exemple ci-après explique d'ailleurs pourquoi les commandes `ALTER OWNER TO` sur les objets sont exécutées deux fois : la commande `ALTER SCHEMA OWNER TO` entraîne la mise à jour du producteur dans la table de gestion, ce qui relance automatiquement la fonction, afin qu'elle modifie les propriétaires des objets en conséquence. Ainsi les quatre premiers messages sont produits par les mécanismes d'ASGARD. Puis l'exécution initiale de la fonction se poursuit, avec une nouvelle exécution des commandes sur les objets...

La fonction `asgard_grant_to_revoke` transforme une commande de type GRANT en son équivalent REVOKE, ou inversement.

SCHÉMA : `z_asgard`.

ARGUMENT : `c_grant` est une chaîne de caractères correspondant à une commande GRANT/REVOKE présumée valide.

RÉSULTAT : une commande de type REVOKE/GRANT (chaîne de caractères).

Cette fonction est un petit utilitaire d'ASGARD, qui sert notamment à préparer la suppression des privilèges des anciens lecteurs et éditeurs lorsque de nouveaux rôles sont désignés. Il n'a en lui-même aucun effet sur les objets ou les données.

EXEMPLE :

```
SELECT z_asgard.asgard_grant_to_revoke('GRANT CREATE ON SCHEMA
c_bibliotheque TO "jon.snow"') ;
```

renvoie

Données	EXPLAIN	Messages	Notifications
asgard_grant_to_revoke			
text			
1	REVOKE CREATE ON SCHEMA c_bibliotheque FROM "jon.snow"		

ASGARD_SYNTHESE_PUBLIC

```
z_asgard.asgard_synthese_public(n_schema regnamespace)
```

La fonction `asgard_synthese_public` renvoie une table contenant une liste de commandes GRANT et REVOKE permettant de recréer les droits du pseudo-rôle public sur le schéma donné en argument et les objets qu'il contient.

Elle ne s'intéresse pas aux objets de type fonction (dont agrégats) et type (dont domaines), sur lesquels public reçoit des droits par défaut qu'il n'est pas judicieux de reproduire sur un autre rôle, ni de révoquer lors d'un changement de lecteur/éditeur. Si des privilèges par défaut ont été révoqués pour public, la révocation restera valable pour les futurs lecteurs/éditeurs puisqu'il n'y a pas d'attribution de privilèges supplémentaires pour les lecteurs/éditeurs sur ces objets.

SCHÉMA : `z_asgard`.

ARGUMENT : `n_schema` est un nom de schéma valide, casté en `regnamespace` (ex : `'w_snow'::regnamespace`).

RÉSULTAT : une table avec un unique champ nommé `commande` et autant d'enregistrements que de privilèges identifiés.

ASGARD_SYNTHESE_PUBLIC_OBJ

```
z_asgard.asgard_synthese_public_obj(obj_oid oid, obj_type text)
```

La fonction `asgard_synthese_public_obj` renvoie une table contenant une liste de commandes `GRANT` et `REVOKE` permettant de recréer les droits du pseudo-rôle public sur un objet de type table, table étrangère, partition de table, vue, vue matérialisée ou séquence.

Elle ne s'intéresse pas aux objets de type fonction (dont agrégats) et type (dont domaines), sur lesquels public reçoit des droits par défaut qu'il n'est pas judicieux de reproduire sur un autre rôle, ni de révoquer lors d'un changement de lecteur/éditeur. Si des privilèges par défaut ont été révoqués pour public, la révocation restera valable pour les futurs lecteurs/éditeurs puisqu'il n'y a pas d'attribution de privilèges supplémentaires pour les lecteurs/éditeurs sur ces objets.

SCHÉMA : `z_asgard`.

ARGUMENTS :

- (1) ***obj_oid*** est l'identifiant PostgreSQL de l'objet à considérer (pour une table, par exemple, peut être obtenu via `'nom_table'::regclass::oid`);
- (2) ***obj_type*** est une chaîne de caractères indiquant le type de l'objet. Valeurs autorisées : « table », « partitioned table », « view », « materialized view », « foreign table », « sequence ».

RÉSULTAT : une table avec un unique champ nommé `commande` et autant d'enregistrements que de privilèges identifiés.

ASGARD_SYNTHESE_ROLE

```
z_asgard.asgard_synthese_role(n_schema regnamespace, n_role regrole)
```

La fonction `asgard_synthese_role` renvoie une table contenant une liste de commandes GRANT et REVOKE permettant de recréer les droits d'un rôle donné sur un schéma et son contenu.

Si le rôle considéré est le producteur du schéma, supposé détenir tous les droits sur celui-ci, ce sont les privilèges révoqués qui sont listés⁵¹ et non les privilèges disponibles.

SCHÉMA : `z_asgard`.

ARGUMENTS :

(1) ***n_schema*** est un nom de schéma valide, casté en `regnamespace` (ex : `'w_snow'::regnamespace`).

(2) ***n_role*** est un nom de rôle valide, casté en `regrole` (ex : `'"jon.snow"'::regrole`).

RÉSULTAT : une table avec un unique champ nommé `commande` et autant d'enregistrements que de privilèges identifiés.

ASGARD_SYNTHESE_ROLE_OBJ

```
z_asgard.asgard_synthese_role_obj(obj_oid oid, obj_type text, n_role
regrole)
```

La fonction `asgard_synthese_role_obj` renvoie une table contenant une liste de commandes GRANT et REVOKE permettant de recréer les droits d'un rôle donné sur un objet de type table, table étrangère, partition de table, vue, vue matérialisée, séquence, fonction (dont fonctions d'agrégation), type (dont domaines).

Si le rôle considéré est le producteur du schéma, supposé détenir tous les droits sur celui-ci, ce sont les privilèges révoqués qui sont listés⁵² et non les privilèges disponibles.

SCHÉMA : `z_asgard`.

ARGUMENTS :

(1) ***obj_oid*** est l'identifiant PostgreSQL de l'objet à considérer (pour une table, par exemple, peut être obtenu via `'nom_table'::regclass::oid`) ;

⁵¹ Des commandes GRANT sont possibles en cas de révocation d'un privilège sur une table (ou équivalent) puis d'octroi de ce même privilège sur un champ.

⁵² Idem.

(2) **obj_type** est une chaîne de caractères indiquant le type de l'objet. Valeurs autorisées : « table », « partitioned table », « view », « materialized view », « foreign table », « sequence », « function », « aggregate », « type », « domain » ;

(3) **n_role** est un nom de rôle valide, casté en regrole (ex : '"jon.snow"'::regrole).

RÉSULTAT : une table avec un unique champ nommé commande et autant d'enregistrements que de privilèges identifiés.

ASGARD_ROLE_TRANS_ACL

```
z_asgard.asgard_role_trans_acl(n_role regrole)
```

La fonction `asgard_role_trans_acl` réécrit les noms de rôles sous la forme d'une expression régulière comparable avec les champs `acl` des tables de `pg_catalog` (par exemple `relacl` de `pg_class`).

Elle est utile aux fonctions `asgard_synthese_role` et `asgard_synthese_role_obj`.

SCHÉMA : `z_asgard`.

ARGUMENTS : **n_role** est un nom de rôle valide, casté en regrole (ex : '"jon.snow"'::regrole).

RÉSULTAT : une chaîne de caractères correspondant à la traduction du nom.

ANNEXE B — LISTE DES RÈGLES, PRINCIPES ET BONNES PRATIQUES

I — INTRODUCTION

Page 8. BONNE PRATIQUE. Les droits sur les objets sont exclusivement accordés aux rôles de groupe.

Page 9. BONNE PRATIQUE. Pour autoriser à un utilisateur à créer des rôles ou des bases, on donne à son rôle de connexion l'attribut CREATEROLE ou, respectivement, l'attribut CREATEDB.

II — GESTION DES DROITS AVEC ASGARD

Page 10. RÈGLE IMPOSÉE PAR ASGARD. Le propriétaire d'un schéma est le propriétaire de tous les objets qu'il contient.

Page 11. Avec ASGARD, tout schéma a un rôle PRODUCTEUR (propriétaire) et au plus un rôle ÉDITEUR et un rôle LECTEUR.

Page 12. BONNE PRATIQUE. Les rôles de connexion individuels sont les identifiants usuels des agents, [prenom].[nom].

Page 13. BONNE PRATIQUE. Les rôles de groupe sont préfixés de g_.

Page 13. RÈGLE. Ne jamais utiliser le caractère « = » dans les noms de rôles. Ne jamais utiliser le caractère « % » dans les noms d'objets, quelle que soit leur nature.

Page 15. BONNE PRATIQUE. g_consult est désigné comme lecteur de tous les schémas contenant des données publiques.

Page 15. BONNE PRATIQUE. g_consult étant identifié comme un rôle de lecture, aucun droit en écriture ne lui est octroyé.

Page 15. BONNE PRATIQUE. Tous les rôles de connexion sont faits membres de g_consult.

Page 16. BONNE PRATIQUE. Seuls le ou les ADL sont membres de g_admin.

Page 17. RÈGLE. Ne pas modifier les permissions de g_admin sur les objets d'ASGARD.

Page 17. BONNE PRATIQUE. Lors de la création d'un rôle de groupe susceptible d'être par la suite identifié comme producteur d'un schéma, rendre immédiatement g_admin membre de ce rôle.

Page 18. BONNE PRATIQUE. consult.default ne dispose d'aucun privilège sur la base si ce n'est l'appartenance à g_consult.

Page 18. BONNE PRATIQUE. Seuls les utilisateurs identifiés par un rôle de connexion personnalisé [prénom].[nom] appartiennent à des groupes ayant des droits en écriture ou un accès en lecture à des données à diffusion restreinte.

Page 19. RÈGLE. Ne jamais modifier les permissions de `g_admin_ext` sur les objets d'ASGARD, ni attribuer ce rôle à quiconque.

III — UTILISATION PRATIQUE D'ASGARD

Page 21. Avec ASGARD, la gestion des droits se fait soit par les commandes habituelles, soit via la vue `gestion_schema_usr` du schéma `z_asgard`.

Page 22. `gestion_schema_usr` ne montre à l'utilisateur que les schémas dont il est habilité à gérer les droits.

Page 23. RÈGLE. Tous les membres de rôles identifiés comme producteurs doivent être membres de `g_consult`.

Page 24. RÈGLE. Pour déléguer la création de schémas à un rôle, il faut lui conférer le privilège `CREATE` sur la base et s'assurer qu'il soit membre de l'éditeur du schéma `z_asgard`.

Page 27. Pour supprimer un schéma via la table de gestion, il faut d'abord changer son bloc en « d » (corbeille) puis basculer `creation` sur `False`.

Page 35. RÈGLE. Le producteur ne peut pas être un rôle de connexion, sauf s'il s'agit d'un super-utilisateur.

Page 45. Lorsqu'un nouvel éditeur ou lecteur est désigné, les mécanismes automatiques d'ASGARD lui transfèrent à l'identique les privilèges de l'ancien, et PostgreSQL fait de même pour les propriétaires.

Page 47. Lorsqu'un objet change de schéma, les privilèges de l'éditeur et du lecteur ne sont pas modifiés en conséquence, sauf à déplacer l'objet avec la fonction `asgard_deplace_obj`.

IV — FONCTIONS UTILITAIRES

V — MENU POUR QGIS

Page 88. BONNE PRATIQUE. S'il est d'usage que les utilisateurs enregistrent dans QGIS leurs identifiants de connexion au serveur PostgreSQL, le paramètre `connex_param` de la table de paramétrage d'ASGARD est modifié par l'ajout du mot-clé `user`.

Page 89. BONNE PRATIQUE. Si, pour les connexions distantes au serveur, un mode SSL autre que `require/requiert` est utilisé, le champ `sslmode` de la table de paramétrage doit être modifié en conséquence.

Page 92. BONNE PRATIQUE. Si, dans QGIS, les connexions au serveur PostgreSQL sont définies à l'aide d'un « service », les paramètres `nom_service` et `connex_param` sont adaptés en conséquence.

VI — INSTALLATION D'ASGARD

Page 99. BONNE PRATIQUE. Après l'installation d'ASGARD, utiliser la fonction `asgard_initialisation_gestion_schema` pour initialiser sa table de gestion.

VII — MISE EN ŒUVRE SUR UN EXEMPLE