ACube

Normes de développement SQL

Version 1.1 du 22/02/2010

Etat : Validé

SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Vérification	Date
1.0	G.Pasquereau	Version initiale		01/04/04
1.1	G;Pasquereau	Actualisation		22/02/10

LISTE DE DIFFUSION

Organisation	Nom	Info	Commentaire	Validation
		☐	☐	☐
		☐	☐	☐
		☐	☐	☐

SOMMAIRE

1 OBJET DU DOCUMENT 4

2 LES REGLES DE NOMMAGE DES OBJETS 5
2.1 – Généralités sur les identifiants 5
2.2 – Les tables 5
2.2.1 Le nom des tables 5
2.2.2 Les colonnes 5
2.2.3 Les type de données utilisateur 6
2.2.4 Les index 6
2.3 Les déclencheurs 7
2.4 Les vues 7
2.5 Les procedures 7
2.5.1 Le nom des procédures 7
2.5.2 Les paramètres 8

3 LES REGLES DE DEVELOPPEMENT 9
3.1 Langage SQL 9
3.2 Procédures stockées 9
3.3 Type des variables 9
3.4 Transactions 10
3.5 Erreur 10
3.6 Jointure 10

TABLEAUX

FIGURES

DOCUMENTS DE REFERENCE

Version	Titre
	Norme SQL ISO/IEC 9075:2003

1 OBJET DU DOCUMENT

Ce document a pour objet de définir les règles de nommage des différentes objets et éléments SQL utilisés par un serveur de base de données.

Il fournit aussi un cadre de recommandations et de bonnes pratiques pour les développements SQL afin de respecter autant que possible la norme ISO/IEC SQL-2003.

2 LES REGLES DE NOMMAGE DES OBJETS

2.1 – GENERALITES SUR LES IDENTIFIANTS

Les identifiants servent à nommer les objets SQL tels que :

- tables,
- procédures,
- triggers…

Les règles syntaxiques générales qui s'appliquent à ces identifiants sont les suivantes :

- La longueur maximale d'un identifiant est de 50 caractères.
- Un nom d'identifiant commence nécessairement par une lettre.
- Les caractères qui peuvent être utilisés sont :
  - les lettres non accentuées,
  - les chiffres,
  - le caractère _ (underscore).

2.2 – LES TABLES

2.2.1 LE NOM DES TABLES

Les seules contraintes existant sur les identifiants des tables sont la conformité aux règles syntaxiques générales des objets SQL. Toutefois, il est recommandé de mettre en œuvre les bonnes pratiques suivantes :

- Le nom de la table doit être court et permettre d'identifier rapidement l'entité fonctionnelle concernée.

  Ex : Poste, Pays, …
- Si le nom de la table est trop long pour une utilisation simple, une abréviation dont la signification reste aisée à déchiffrer peut être utilisée.

  Ex : ImpBud pour Imputation Budgétaire
- Il est préférable, dans le cas d'un nombre important de tables, d'utiliser un préfixe de 3 caractères suivi du caractère souligné faisant référence à un domaine fonctionnel particulier.

  Ex : prv_budget pour budget prévisionnel, rea_budget pour budget réalisé.

2.2.2 LES COLONNES

- Le nom d'une colonne de table doit être court et permettre d'identifier rapidement l'information.

  Ex : Nom, Adresse, …

- Le type de données de la colonne doit correspondre de préférence à un type de données utilisateur (cf § 2.2.3).
- Chaque table possède une colonne identifiant considérée comme clé primaire. Son nom est défini sous la forme :

<id>_<NomDeLaTable>

où :

<id> : désigne les caractères « id »,

<NomDeLaTabble> : représente le nom de la table.

Ex : id_pays est la colonne identifiant de la table pays.

2.2.3 LES TYPE DE DONNEES UTILISATEUR

Les types de données utilisateurs (UDT) permettent à codifier de façon personnalisée des type de données standards. Les UDT permettent donc d'assurer une plus grande homogénéité entre des colonnes de table de nature similaire et de faire ainsi évoluer plus aisément un schéma de données.

Il n'y a pas de contraintes particulières sur la désignation de ces UDT qui devront néanmoins respecter les quelques règles suivantes :

- La désignation doit être significative et correspondre à une réalité fonctionnelle.

  Ex : telephone, libelle_court, libelle_long, …
- La désignation ne doit pas faire référence directement aux type de données et/ou à la taille du type de données standard sur lequel il s'appuie.

  Ex : pas de noms tels que alphanum2 ou decimal3.

2.2.4 LES INDEX

Les identifiants des index de table sont définis sous la forme :

<Préfixe>_<NomDeLaTable>_<TypeIndex>_<LibelléIndex>

où :

<Préfixe> :	I (Initiale de Index),
<NomDeLaTable> :	Nom de la table à laquelle est rattaché l'index,
<TypeIndex> :	PK dans le cas d'un index de clé primaire,
	FK dans le cas d'un index de clé étrangère,
	AK dans le cas d'un index de clé alternative,
	XK dans les autres cas (index de performance, …),
<LibelléIndex> :	Intitulé de l'index (de préférence le nom de la colonne sur laquelle est appliqué l'index).

Ex : i_personne_ AK_nom est l'index de clé alternative sur la colonne nom de la table personne.

2.3 LES DECLENCHEURS

Les déclencheurs permettent de lancer automatiquement des traitements lors de la mise à jour des informations d'une table.

Les identifiants des déclencheurs sont définis sous la forme :

<Préfixe>_<NomDeLaTable>_<Traitement>[_<LibelléTrigger>]

où :

<Préfixe> :	T (Initiale de Trigger),
<NomDeLaTable> :	Nom de la table à laquelle est rattaché le déclencheur,
<Traitement> :	I dans le cas d'une insertion,
	U dans le cas d'une mise à jour,
	D dans le cas d'une suppression,
< LibelléTrigger > :	Intitulé du déclencheur.

Ex : t_personne_iu est un déclencheur en insertion et mise à jour sur la table personne,

t_pays_u_designation est un déclencheur en mise à jour sur la table pays.

2.4 LES VUES

Une vue peut être représentée comme une table virtuelle dont les données ne sont pas stockées dans la base sous forme d'objet distinct.

Les identifiants des vues sont définis sous la forme :

<Préfixe>_<NomDeLaVue>

où :

<Préfixe> :	V (Initiale de Vue),
<NomDeLaVue> :	Libellé de la vue permettant de l'identifier de façon claire.

Ex : v_poste_pays est un exemple de vue (sur les tables pays et poste).

2.5 LES PROCEDURES

2.5.1 LE NOM DES PROCEDURES

Les identifiants des procédures sont définis sous la forme :

<Préfixe>_<Traitement>_<NomDeLaProcedure>

où :

<Préfixe> :	P (Initiale de Procédure),
<Traitement> :	S dans le cas d'une sélection simple,
	M dans le cas d'une mise à jour de données,

A dans les autres cas,

<NomDeLaProcedure> : Libellé de la procédure précisant le traitement.

Ex : p_s_pays liste la table pays,
P_m_dossier met à jour un dossier.

2.5.2LES PARAMETRES

Les noms des paramètres des procédures sont définis sous la forme :

<Préfixe><Type>_<NomDuParametre>

où :

<Préfixe> : A (Initiale de Argument)

<Type> : I dans le cas d'un paramètre en entrée,
O dans le cas d'un paramètre en sortie,
IO dans le cas d'un paramètre en entrée/sortie,

< NomDuParametre > : Intitulé du paramètre de la procédure.

Ex : ai_codepays est un paramètre en entrée,
aio_accord est un paramètre en entrée/sortie met à jour un dossier.

3 LES REGLES DE DEVELOPPEMENT

3.1 LANGAGE SQL

Par soucis de portabilité, il faut privilégier l'utilisation de la norme ISO/IEC SQL-2003 par rapport à l'implémentation propriétaire de l'éditeur.

3.2 PROCEDURES STOCKEES

Dans le cadre de l'architecture ACube, l'utilisation des procédures stockées doit être strictement cantonnée à l'accès et à la mise à jour « complexe » de l'information.

Cette recommandation implique que :

- Aucune règle de gestion fonctionnelle ne doit figurer au sein de procédures stockées.
- Les lectures simples (sans jointure ou avec des jointures simples) ne sont pas mises en œuvre par des procédures stockées.
- Les mises à jour (insert, update, delete) unitaires de tables ne sont pas non plus mises en œuvre par des procédures stockées.

3.3 TYPE DES VARIABLES

Chaque éditeur de serveur SQL de base de données propose, en complément de la liste des types de données définis dans la norme SQL, un ensemble de type de données propriétaire.

Dans un souci de portabilité et de respect de normes de développement, il est donc nécessaire d'utiliser prioritairement les types suivants définis dans la norme ISO/IEC SQL-2003 (ou leurs équivalents dans l'implémentation de l'éditeur) :

- Character,
- Character varying,
- Character large object,
- Binary large object,
- Numeric,
- Decimal,
- Smallint,
- Integer,
- Bigint,
- Float,
- Real,
- Double precision,

- Boolean,
- Date,
- Datetime,
- Timestamp,
- Interval.

3.4 TRANSACTIONS

En architecture multi-niveaux, la notion de transaction qui garantit la cohérence d'un traitement de mise à jour ne doit pas être pris en charge au sein d'une procédure stockée.

Celle-ci n'étant qu'un élément parmi d'autres du processus de mise à jour de l'information, la transaction et son bon déroulement sont initiés et gérés par le serveur d'application.

3.5 ERREUR

Chaque procédure stockée doit retourner une valeur retour au processus appelant :

Valeur retournée = 0 : Le traitement s'est correctement déroulé,
<> 0 : Le traitement ne s'est pas correctement déroulé.

3.6 JOINTURE

Les jointures internes et externes de tables doivent utiliser la syntaxe la syntaxe ISO/IEC SQL-2003 « INNER JOIN » et « OUTER JOIN » dans la clause FROM.

Les jointures s'appuyant sur la clause WHERE sont interdites.

Exemple de jointure interne :

```
SELECT *
FROM tab1 INNER JOIN tab2
ON tab1.c3 = tab2.c4
```

Exemple de jointure externe :

```
SELECT *
FROM tab1 LEFT OUTER JOIN tab2
ON tab1.c3 = tab2.c3
```