Normalisation des noms des objets des bases de données

Sommaire

Cet article a pour but de vous proposer des méthodes et une normalisation interne de l'ensemble des éléments composant une base de données.
Une normalisation interne (règle propre à l'entreprise) a beaucoup d'intérêt en matière de développement informatique : elle facilite le travail de tous, en équipe, comme le travail solitaire. Elle permet de repérer facilement la nature et le contenu d'un objet par son simple nom évitant ainsi d'établir un descriptif très complet de l'ensemble des composants d'une base de données. Enfin, elle optimise l'exécution des procédures et requêtes.
Des règles simples et de bon sens sont un gage d'application facile à écrire, bien développée et surtout PORTABLE d'un environnement l'autre ou d'un SGBDR à l'autre.

L'auteur

SQLPro 

L'article

Publié le 26 août 2003

Public visé : intermédiaire

1. Vocabulaire

- Une **entité** est un ensemble de données cohérents ayant des caractéristiques communes (**PERSONNE**, **VEHICULE**...)
- Une **relation** (ou **association**) représente un moyen de relier, souvent par une action deux ou plus de deux entités (personne **CONDUIT** véhicule)
- Un **attribut** est une propriété ou caractéristique qualifiant l'entité (personne.**SEXE**, vehicule.**IMMATRICULATION**). Il possède un nom et un type de donnée
- Un **domaine** est l'ensemble des valeurs que pourra prendre l'attribut. (sexe = [**HOMME**, **FEMME**])
- La **cardinalité** est le nombre possible des liens d'une relation (personne conduit **0** ou **1** véhicule, personne possède **0** ou **N** véhicule)
- Un **identifiant** est un attribut ou un ensemble d'attribut permettant d'identifier de façon unique une occurrence de l'entité (le **n° de sécurité sociale 1600475112335** permet d'identifier de manière unique une personne)
- Une clé étrangère est un attribut d'une relation qui fait référence à la clé d'une autre relation afin de: c'est ainsi que l'on pourra lier plusieurs relations

2. Considération générales

2.1. Outil de modélisation

Toute base de données sera modélisé par un outil externe au produit cible permettant :

- une modélisation conceptuelle (modèle logique) et physique (organisation des données)
- le choix de divers et nombreux SGBDR cibles
- utilisant une méthode connue et répandue comme MERISE, E/R ou UML

Voici une liste non exhaustive de plusieurs solutions d'atelier de modélisation :

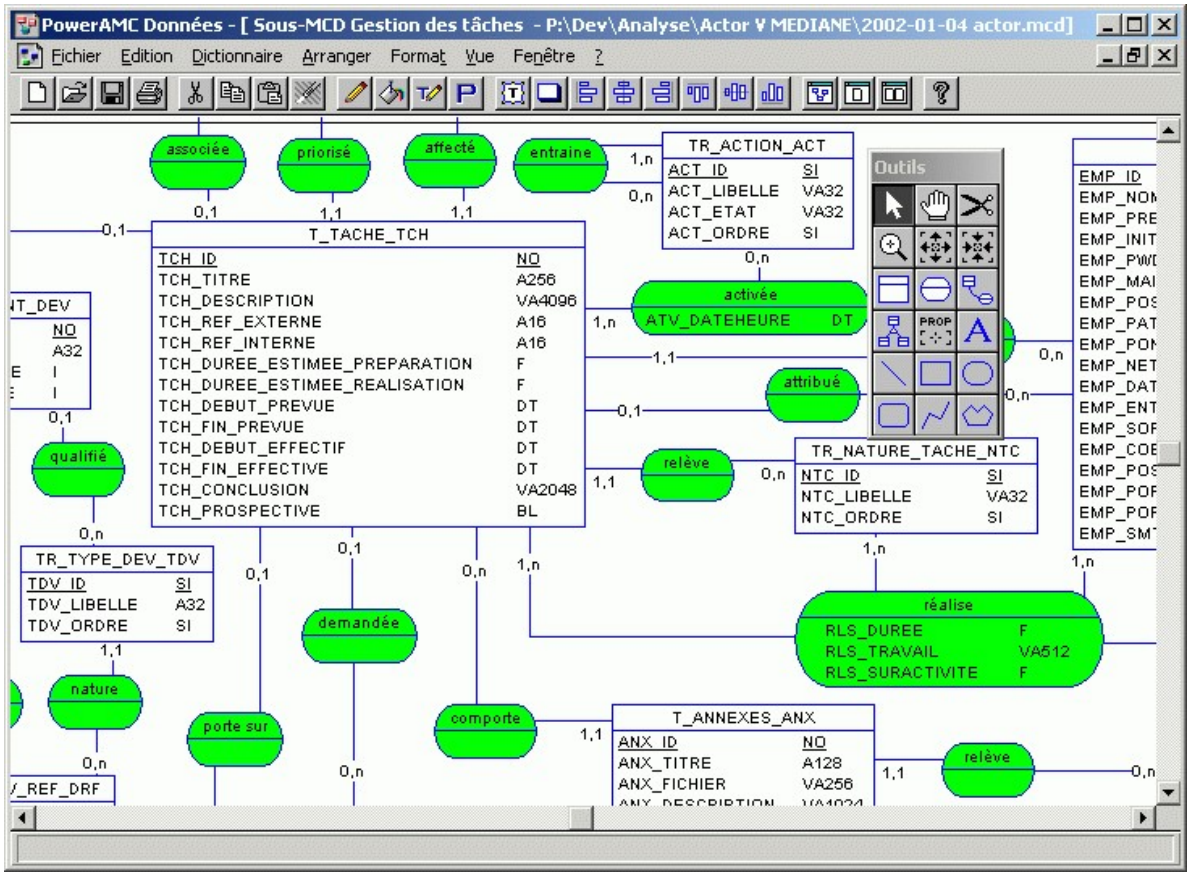
DataBase Design Studio de Chili Source Software	http://www.chillisource.com/
DeZign for DataBases de Datanamic	http://www.datanamic.com/
ERWin de Computer Associates	http://www3.ca.com/Solutions/Product.asp?ID=260
Mega Suite de Mega International	http://www.mega.com/
PowerDesigner de PowerSoft (ex AMC*Designor)	http://www.sybase.com/products/enterprisemodeling/powerdesigner
Win'Design de Cecima	http://www.win-design.com/en/index.htm
xCase de Resolution	http://www.xcase.co.il/
DataArchitect de theKOMPANYcom	http://www.thekompany.com/products/dataarchitect/
Rational Rose de Rational Software	http://www.rational.com/
Case Studio 2 de CharonWare	http://www.casestudio.fr
ER Studio de Embarcadero	http://www.embarcadero.com/
Visio de Microsoft (ex infomodeler de Synactics)	http://www.microsoft.com/office/visio/
Silverrun de Magna solutions	http://www.silverrun.com/
System Architect de Popkin	http://www.popkin.com/products/sa2001/systemarchitect.htm
Designer 2000 d'Oracle (mono base)	http://www.oracle.com/ (???)
Casewise	http://www.casewise.com/
QuickUML de Excel (Linux et Windows) de Excel Software	http://excelsoftware.com/quickumllinuxnews100.html
MacA&D / WinA&D de Excel Software	http://excelsoftware.com/richdatamodel.html
Select d'Aonix	http://www.aonix.com/

Les poids lourds sont : Power Designer (ex AMC Designor), ER Win, et Rational Rose.

Enfin, à titre d'information, voici quelques liens concernant les méthodes et outils de modélisation en tout genre :

- http://www.cs.queensu.ca/Software-Engineering/toolcat.html
- http://www.aisintl.com/

Exemple d'écran de Power Designer montrant un Modèle Conceptuel de Données MERISE :



2.2. Normalisation

Les modèles seront devront répondre obligatoirement des trois premières formes normales (1NF, 2NF, 3NF).
Dans la mesure du possible on respectera les formes normales de Boyce Codd (BCNF) et les 4eme et 5eme formes normales à condition que cela n'entraîne pas plus d'inconvénient pour l'écriture des requêtes et des traitements que d'avantages.

Rappels :

Sélectionnez 1NF Première forme normale	Sélectionnez Une relation est en première forme normale si et seulement si : - tout attribut contient une valeur atomique. * Chaque entité doit disposer d'un identifiant qui la caractérise de manière unique).
Sélectionnez 2NF Deuxième forme normale	Sélectionnez Une relation est en deuxième forme normale si et seulement si : - elle est en 1NF, - tout attribut n'appartenant pas à une clé ne dépend pas que d'une partie de cette clé. * Les propriétés d'une entité ne doivent dépendre que de l'identifiant de l'entité et non d'une partie de cet identifiant
Sélectionnez	Sélectionnez

3NF Troisième forme normale	Une relation est en troisième forme normale si et seulement si : - elle est en 2NF, - tout attribut n'appartenant pas à une clé ne dépend pas d'un attribut non clé. * Les propriétés d'une entité doivent dépendre de l'identifiant de l'entité de manière directe
Sélectionnez BCNF Forme normale de BOYCE-CODD	Sélectionnez Une relation est en Forme normale de BOYCE-CODD (BCNF) si, et seulement si : - elle est en 3NF, - et seules les seules dépendances fonctionnelles élémentaires qu'elle comporte sont celles dans lesquelles une clé détermine un attribut. * Pour les identifiants composés de plusieurs propriétés, ces dernières ne doivent pas être dépendantes d'une autre propriété de l'entité.
Sélectionnez 4NF Quatrième forme normale	Sélectionnez Une association est en 4FN si et seulement si : - elle est en BCFN - lorsqu'il existe une dépendance multivaluée élémentaire, celle-ci est unique. * Toute relation a une décomposition en quatrième forme normale qui est sans perte. Cette décomposition n'est pas forcément unique.
Sélectionnez 5NF Cinquième forme normale	Sélectionnez Une association est en 5FN si et seulement si : - elle est en 4FN - et si elle ne possède pas de DJ ou bien toute dépendance de jointure est impliquée par des clés candidates de R. * La cinquième forme normale est une généralisation de la quatrième forme normale qui nécessite de prendre en compte les dépendances de jointure induites par la connaissance des clés d'une relation

On trouvera des compléments d'information sur les formes normales aux URL suivantes.

- http://philippe.guezelou.free.fr/mcd/mcd.htm
- http://www.bd.enst.fr/polyv7/chap5.htm
- http://tcosnuau.free.fr/COURS/MODREL/MODREL.HTM

3. Nommage des objets

Les noms des objets d'un modèle ou d'un schéma devront être significatifs et pertinents. Ils devront être consitué uniquement des caractères suivants :

Sélectionnez

```
[a .. z] + [A .. Z] + [0 .. 9] +[ _ ]
```

Avec les restrictions suivantes :

- ne pas dépasser 128 caractères
- ne doit pas commencer par un chiffre
- ne peut avoir plusieurs caractères "blanc souligné" de suite
- la casse n'a pas d'importance
- le nom ne doit pas être un mot réservé de SQL (voir à ce sujet : LES MOTS CLEFS DU SQL)

ATTENTION : Autrement dit, les lettres accentuées (é à ù ï É ...), les "kanas" (ç œ ...), les caractères de ponctuation (, ; : ! ? ...) et autres caractères spéciaux, comme le blanc, sont proscrits.

Exemples :

_TOTO	Autorisé
123TOTO	Interdit
TITI__TATA	Interdit
_toto	Autorisé (mais identique au premier)
Vérité	Interdit

ATTENTION :
On réservera les noms en minuscule aux modèles logiques et les noms en majuscule aux modèles physique.

Exemple :

client	nom logique (entité par exemple)
T_CLIENT_CLI	nom physique (table par exemple)

et leur longueur devra tenir compte des limites du nombre de caractères utilisables dans le nom des objets du SGBDR et des variables du langage de programmation interne au SGBDR (triggers et procédures stockées).

3.1. Nom d'un serveur

Le nom d'un serveur doit commencer par le préfixe SRV_ suivi d'une indication pertinente.

Exemples :

SRV_GESTION	pour un serveur de base de données de gestion
SRV_FO	pour un serveur "front office"

3.2. Nom d'une base de donnée

Le nom d'une base de donnée doit commencer par le préfixe BD_ suivi d'une indication pertinente.

Exemples :

BD_GESCOM	Base de données de GEStion COMmerciale
BD_SUIVIPROD	Base de données de SUIVI de PRODuction

3.3. Domaines

Un domaine doit toujours commencer par le préfixe D_ suivi d'une lettre indiquant la famille du type parmi les éléments suivants :

D_A_	Domaine de famille chaîne de caractères
D_B_	Domaine de famille binaire
D_N_	Domaine de famille numérique
D_T_	Domaine de famille temporel

Il doit être suivi d'un nom indiquant l'usage.

Exemples :

D_N_POURCENT	Réel compris entre 0.0 et 100.0
D_N_ID	Numérique d'identification de type entier
D_A_ADRESSE	Chaîne de caractère varchar(32)
D_B_BOOLEEN	Booléen

Le mieux étant de partir d'une liste de domaine par défaut que l'on reprend systématiquement pour chaque base.
Par exemple :

Sélectionnez

```
CREATE DOMAIN  D_A_ADRESSE as VARCHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_CAR as CHAR(1)
```

Sélectionnez

```
CREATE DOMAIN  D_A_CODE as CHAR(8)
```

Sélectionnez

```
CREATE DOMAIN  D_A_CODE_COURT as CHAR(4)
```

Sélectionnez

```
CREATE DOMAIN  D_A_CODE_LONG as CHAR(16)
```

Sélectionnez

```
CREATE DOMAIN  D_A_CODE_POSTAL as CHAR(8)
```

Sélectionnez

```
CREATE DOMAIN  D_A_LIB as VARCHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_LIB_COURT as VARCHAR(16)
```

Sélectionnez

```
CREATE DOMAIN  D_A_LIB_FIXE_COURT as
CHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_LIB_FIXE_LONG as
CHAR(64)
```

Sélectionnez

```
CREATE DOMAIN  D_A_LIB_LONG as VARCHAR(64)
```

Sélectionnez

```
CREATE DOMAIN  D_A_MAIL as VARCHAR(128)
```

Sélectionnez

```
CREATE DOMAIN  D_A_NOM as CHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_PRENOM as VARCHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TEL as CHAR(20)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TEXTE as VARCHAR(2000)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TEXTE_BLOB as TXT
```

Sélectionnez

```
CREATE DOMAIN  D_A_TEXTE_COURT as
VARCHAR(1000)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TEXTE_LONG as
VARCHAR(4000)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TITRE as VARCHAR(128)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TITRE_COURT as
VARCHAR(64)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TITRE_LONG as
VARCHAR(256)
```

Sélectionnez

```
CREATE DOMAIN  D_A_VILLE as CHAR(32)
```

Sélectionnez

```
CREATE DOMAIN  D_A_TRIGRAMME as CHAR(3)
```

Sélectionnez

```
CREATE DOMAIN  D_B_BOOLEEN as BIT(1)
```

Sélectionnez

```
CREATE DOMAIN  D_B_CODE_COULEUR
VARBINARY(6)
```

Sélectionnez

```
CREATE DOMAIN  D_N_DECIMAL as DECIMAL(16,2)
```

Sélectionnez

```
CREATE DOMAIN  D_N_FLOAT as FLOAT
```

Sélectionnez

```
CREATE DOMAIN  D_N_INT_COURT as SMALLINT
```

Sélectionnez

```
CREATE DOMAIN  D_N_INT_LONG as INTEGER
```

Sélectionnez

```
CREATE DOMAIN  D_N_NID as INTEGER VALUES >
0
```

Sélectionnez

```
CREATE DOMAIN  D_N_POURCENT as FLOAT VALUES
BETWEEN 0 AND 100
```

Sélectionnez

```
CREATE DOMAIN  D_T_DATE as DATE
```

Sélectionnez

```
CREATE DOMAIN  D_T_DATEHEURE as DATE
```

Sélectionnez

```
CREATE DOMAIN  D_T_HEURE as TIME
```

3.4. Nom d'une entité

Elle doit commencer par un préfixe :

e_	lorsqu'il s'agit d'entité fonctionnelle
er_	lorsqu'il s'agit d'entité de référence
es_	lorsqu'il s'agit d'entité "système"

ATTENTION : L'emploi du pluriel est à proscrire.

Exemples :

e_client	entité fonctionnelle des clients
er_pays	entité de référence des pays
es_user	entité système des utilisateurs

3.5. Nom d'une relation (association)

Elle doit commencer par le préfixe R :

Exemple :

r_loue	relation "loue" entre entité e_client et e_maison
r_achete	relation achète entre e_client et e_maison

3.6. Nom d'un attribut

Le nom d'un attribut doit être suffisament pertinent pour que l'on puisse comprendre la nature du type de données qu'il représente.

Exemple :

nom	attribut nom
date_naissance	attribut date de naissance
quantite	attribut quantité

Dans la mesure du possible on évitera les noms par trop génériques. Ainsi il conviendra de proscrire : "numero", "date", "type"...

3.7. Nom d'une table, d'une vue

Elle doit commencer par un préfixe :

T_	lorsqu'il s'agit d'une table fonctionnelle	V_
TR_	lorsqu'il s'agit d'une table de référence	VR_
TS_	lorsqu'il s'agit d'une table "système"	VS_
TJ_	lorsqu'il s'agit d'une table de jointure	VJ_
TG_	lorsqu'il s'agit d'une table générique (héritage)	VG_

Elle doit reprendre le corps du nom de l'entité, où à défaut (table de jointure) le nom de la relation si cette dernière est nommée.
Elle doit être suffixée par un **trigramme unique** au sein de la base de données, permattant l'identification rapide de la table.

Exemples :

T_CLIENT_CLI	table fonctionnelle des clients
TR_PAYS_PAY	table de référence des pays
TS_USER_USR	table système des utilisateurs
TJ_ACHETE_ACH	table de jointure

	relation "achete"

Une vue sera systématiquement préfixée par V_.

3.8. Nom, ordre de création et taille des colonnes

3.8.1. Ordre de création

L'ordre de création et de description des colonnes devra répondre aux règles suivantes :

- Les colonnes les plus significatives et les plus utilisés seront situées en tête de la description
- Les colonnes les moins fréquemment modifiées ou consultées seront situées en fin de la description
- Les colonnes composant la clef primaire de la table devront être les premières colonnes décrites de la table
- Les colonnes composant les clefs étrangères devront être les suivantes
- Les colonnes doivent être regroupés lorsqu'ils font partie d'un sous-ensemble significatifs de la table

3.8.2. Noms

Les noms de colonnes doivent être préfixée par le trigramme de la table d'origine et reprendre le nom d'attribut.

Exemple :

CLI_NOM	colonne nom de la table T_CLIENT
CLI_DATE_NAISSANCE	colonne date de dnaissance de la table T_CLIENT
FAC_DATE_EMISSION	colonne date d'émission de la table facture

Certaines abréviation d'usage courant devront être utilisées :

ID	identifiant (en général clef auto incrémentée)
NUM	numéro
REF	référence
CODE	code, codage, codification
LIB	libellé
ORD	ordre
CP	code postal
ADR	adresse
FAX	télécopie
MAIL	e-mail
TEL	téléphone
GSM	téléphone mobile
LOG	"login"
PWD	"password"
NBR	nombre

QTE	quantité
POS	position
NDX	index
MNT	montant
TX	taux
PCT	pourcentage
PUTC	prix unitaire toutes taxes
PUHT	prix unitaire hors taxes

3.8.3. Tailles des données des colonnes

En principe la taille (et le format des données) seront libre, et doivent suivre les spécification de l'application. Cependant pour certains champs les limites seront les suivantes :

- Nom d'une personne 36 caractères
- Prénom d'une personne 32 caractères
- Nom d'une ville 32 caractères
- Ligne d'adresse 32 caractères, 4 lignes maximum (normalisation LA POSTE)
- Ville 32 caractères (normalisation LA POSTE)
- Titre d'une personne 5 caractères (abréviations : "M." - "Mme." - "Mlle."- "Me." pour Maître - "Dr." pour Docteur, etc...)
- code pays 3 caractères (codification internationale)

REMARQUE : il y aura lieu de rechercher systématiquement les formats des codifications internationale (normes ISO), des codifications françaises (normes AFNOR) et des codifications par branches de métiers (exemple normes NOEMIE et IRIS inter régime, instruction M21 pour le domaine de la santé publique).

NOTA, EN CE QUI CONCERNE LES ADRESSES : les formats donnés sont ceux de la poste française. Il y a lieu de s'y tenir dans tous les cas. La poste admet jusqu'à 4 lignes d'adresses + une ligne concernant le code postal accompagné du nom de la ville. En outre l'adresse doit être précédée en principe de nom de la personne et/ou du nom de l'organisation (sté., association, syndicat…). Dans le cas contraire, les envois en nombre ne pourront être lus par des machines automatiques et l'économie non réalisée se chiffre en milliers d'euros.

CONSEIL : en matière d'adresse il est important de spécifier le cedex dans une case "à part" afin de pouvoir faciliter la mise au point de requêtes portant sur les villes.

Ainsi une bonne formalisation d'adresse devrait répondre à la description suivante :

CLI_ADR1	Sélectionnez `VARCHAR(32)`
CLI_ADR2	Sélectionnez `VARCHAR(32)`
CLI_ADR3	Sélectionnez `VARCHAR(32)`
CLI_ADR4	

	Sélectionnez `VARCHAR(32)`
CLI_VILLE	Sélectionnez `VARCHAR(32)`
CLI_CP	Sélectionnez `VARCHAR(8) - pour accepter les CP étranger`
CLI_CEDEX	Sélectionnez `VARCHAR(32)`
CLI_PAYS	Sélectionnez `VARCHAR(3) - codification internationale`

3.9. Contraintes de table

Les contraintes de table devront être préfixées C_ suivi d'un indicateur du type de contrainte. Elles seront suffixées par le trigramme de table.

Indicateur de type de contrainte :

C_PK_	contrainte de table "clef primaire"
C_FK_	contrainte de table "clef étrangère"
C_UNI_	contrainte de table "unicité"
C_CHK_	contrainte de table "validité"

- Les contraintes de clef primaire doivent reprendre le trigramme de la table d'origine
- Les contraintes de clef étrangère doivent reprendre le trigramme de la table d'origine et celui de la table dans laquelle elles figurent
- Les contraintes d'unicité et de validité doivent avoir un nom significatif

Exemples :

C_PK_CLI	contrainte de clef primaire de la table client
C_FK_FAC_CLI	contrainte de clef étrangère de la table facture dans la table client
C_UNI_LOGIN_PASSWORD_CLI	contrainte d'unicité pour les colonnes LOGIN et PASSWORD
C_CHK_CP_CLI	contrainte de validité d'un code postal

3.10. Index

Les index doivent être préfixés X_ suivi d'un indicateur de la nature de l'index et d'un nom significatif. Ils seront suffixées par le trigramme de table.

On pourra choisir comme indicateur de nature, parmi les abréviations suivantes :

X_CSR_	index en cluster
X_BMP_	index "bitmap"
X_BTR_	index arbre équilibré
X_HCG_	index, "clef de hachage"

Exemples :

X_CSR_ID_CLI	index clusterisé de l'indentifiant du client
X_BTR_NOM_PRENOM_CLI	index arbre equilibré pour nom/prenom de client
X_HCG_TEL_CLI	index en clef de hachage pour téléphone client

3.11. Procédures stockées

Les procédures stockées seront préfixées par SP_ suivi d'un nom significatif.

Exemple :

SP_CALC_TARIF	procédure stockée de calcul des tarifs

4. Documentation, ergonomie, et écriture

Les règles ci dessous établissent la manière dont les développeurs doivent écrire les requêtes et le code afférent aux objets d'une base de données.

4.1. Écriture des requêtes

Chaque clause de requête devra être indentée :

Sélectionnez

```
MAUVAIS !!!

SELECT CLI_ID, CLI_NOM, CLI_ENSEIGNE,
CLI_PRENOM  FROM T_CLIENT WHERE
CLI_ADR_PAYS = 'F'
AND CLI_ENSEIGNE IS NULL OR CLI_ENSEIGNE
=''
ORDER BY CLI_NOM
```

Sélectionnez

```
BON !!!

SELECT CLI_ID, CLI_NOM, CLI_ENSEIGNE,
       CLI_PRENOM
FROM   T_CLIENT
WHERE  CLI_ADR_PAYS = 'F'
       AND CLI_ENSEIGNE IS NULL OR
CLI_ENSEIGNE =''
```

	ORDER BY CLI_NOM

Toutes les colonnes renvoyées devront être nommées :

Sélectionnez MAUVAIS !!! SELECT CLI_ID, CLI_PRENOM || ' ' || CLI_NOM, CLI_ENSEIGNE FROM T_CLIENT	Sélectionnez BON !!! SELECT CLI_ID, CLI_PRENOM || ' ' || CLI_NOM AS NOM_COMPLET_CLI, CLI_ENSEIGNE FROM T_CLIENT

Les noms des colonnes renvoyées ne doivent pas comporter de doublons (en particulier l'usage de l'étoile dans la clause SELECT est à proscrire) :

Sélectionnez MAUVAIS !!! SELECT * FROM T_CLIENT_CLI CLI INNER JOIN T_FACTURE_FAC FAC ON CLI.CLI_ID = FAC.CLI_ID WHERE CLI.CLI_ADR_PAYS = 'F'	Sélectionnez BON !!! SELECT CLI.CLI_ID, CLI.TIT_CODE, CLI.CLI_NOM, CLI.CLI_PRENOM, CLI.CLI_ENSEIGNE, FAC.FAC_ID, FAC.PMT_CODE, FAC.FAC_DATE, FAC.FAC_PMT_DATE FROM T_CLIENT_CLI CLI INNER JOIN T_FACTURE_FAC FAC ON CLI.CLI_ID = FAC.CLI_ID WHERE CLI.CLI_ADR_PAYS = 'F' -- un seule fois la colonne CLI_ID
	Sélectionnez BON !!! SELECT CLI.CLI_ID AS CLI_CLI_ID, FAC.CLI_ID AS FAC_CLI_ID, CLI.TIT_CODE, CLI.CLI_NOM, CLI.CLI_PRENOM, CLI.CLI_ENSEIGNE, FAC.FAC_ID, FAC.PMT_CODE, FAC.FAC_DATE, FAC.FAC_PMT_DATE FROM T_CLIENT_CLI CLI LEFT OUTER JOIN T_FACTURE_FAC FAC ON CLI.CLI_ID = FAC.CLI_ID WHERE CLI.CLI_ADR_PAYS = 'F' -- deux fois la colonne CLI_ID avec deux noms différents

4.2. Écriture du code

Si l'éditeur texte du code en offre la possibilité, on utilisera toujours une police a espacement fixe telle que la police Courrier.

Le code devra être écrit exclusivement en majuscule, sauf contrainte particulière. Cette règle s'explique dans le sens ou il est important dans un langage hôte de pouvoir faire la distinction entre le code exécuté sur le poste client et le code envoyé et exécuté sur le serveur.

Le renommage des tables dans les requêtes se fera à l'aide du trigramme.
En cas de présence de plusieurs instances de la même table en ajoutera un numéro.

Les différentes clauses et partie de clauses des requêtes devront être indentées avec un retrait d'au moins 3 caractères par type d'item. De même si dans les requêtes figure un branchement CASE.

Les jointures de table devront toujours être réalisées avec l'opérateur JOIN lorsque celui ci est disponible.

Exemple :

Sélectionnez

```
SELECT CLI.CLI_ID, FAC1.FAC_ID,
       CASE FAC1.FAC_MODE_PAIEMENT
            WHEN 'B' THEN 'Chèque bancaire'
            WHEN 'E' THEN 'Espèces'
            WHEN 'C' THEN 'Carte de paiement'
ELSE ''
       END AS FAC_MODE_PAIEMENT
FROM   T_CLIENT_CLI CLI
       INNER JOIN T_FACTURE_FAC FAC1
             ON CLI.CLI_ID = FAC1.CLI_ID
WHERE  CLI.CLI_ADR_PAYS = 'F'
GROUP  BY FAC1.FAC_ID, CLI.CLI_ID
HAVING SUM(FAC1.FAC_MONTANT_TTC) > (SELECT MAX(FAC2.FAC_MONTANT_TTC)
                                    FROM   T_FACTURE_FAC FAC2
                                    WHERE  FAC2.CLI_ID = CLI.CLI_ID)
ORDER BY CLI.CLI_ID, FAC_MODE_PAIEMENT
```

Dans les procédures stockées, l'indentation sera faite :

- pour les requêtes comme indiqué précédemment
- pour les blocs de code, par un retrait pour chaque bloc d'au moins 3 caractères

On veillera en outre à placer des commentaires courts de manière judicieuse.

Exemple :

Sélectionnez

```
CREATE PROCEDURE SP_DEV_SUPPRESSION
       @id_element integer,
       @recursif   bit
AS

DECLARE @OK integer
DECLARE @bg_element integer
DECLARE @bd_element integer
DECLARE @intervalle integer

SET NOCOUNT ON

-- démarrage transaction
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRANSACTION DELETE_TREE

-- vérifie de l'existence de l'élément
SELECT @OK = count(*)
FROM   T_DEVELOPPEMENT_DEV
WHERE  DEV_ID = @id_element
```

```
-- si élément supprimé, alors retour sans insertion avec valeur -1
IF @OK = 0
BEGIN
   SELECT -1
   ROLLBACK TRANSACTION DELETE_TREE
   RETURN
END
...
```

4.3. Cartouche

Toute procédure stockée, trigger ou vue devra être pourvue d'un cartouche dans le code. Le cartouche devra contenir les éléments suivants :

- le nom du développeur ayant codé l'objet
- la date de réalisation
- Un commentaire succinct
- la référence de la nomenclature si une nomenclature des éléments de code a été établie
- la description des paramètres
- s'il y a lieu, les modifications successives apportées au code (identifiant du codeur, date et nature)

Exemples :

Sélectionnez

```
------------------------------------------------------
-- ACTOR V 3 - ARBODEV - GA01
------------------------------------------------------
-- Alfred DURAND - 2001-02-17
------------------------------------------------------
-- Gestion de l'arborescence : insertion d'un fils ainé
-- retourne 0 en cas d'anomalie
-- PARAMETRES :
--    en entrée :
--       id du père : DEV_ID (integer)
--    en sortie :
--       id du fils inseré : DEV_ID (integer)
------------------------------------------------------
-- MODIFICATION
-- 2001-05-11 #1 - Martine DUPONT [MDP] :
--    bug lors du calcul si dd = df
-- 2001-05-11 #2 - Martine DUPONT [MDP] :
--    bug lors du calcul si dd > df
------------------------------------------------------
```

Sélectionnez

```
/* ================================== */
/* GESTION DE L'ARBRE DU DÉVELOPPEMENT */
/* suppression                       */
/* ================================== */
/* Frédéric BROUARD    20/11/2001     */
/* ================================== */
/* suppression d'un élément ou d'un   */
/* sous arbre                         */
/* ================================== */
/* paramètres :                       */
/*   id_element : integer             */
/*   recursif   : bit                 */
/* Si @recursif = 0 =>                */
```

```
/*    supression d'un élément          */
/* si @recursif = 1 =>                 */
/*    supression d'un sous arbre       */
/* ================================== */
```

4.4. Valeur de retour

Dans la mesure du possible une procédure renverra toujours une valeur de retour sous la forme d'un entier, permettant de connaître l'état d'exécution de la procédure.

En cas de succès de la procédure, cette valeur de retour sera 0.
En cas de problème cette valeur sera :

- une valeur négative en cas d'exception (erreur)
- une valeur positive pour des valeur limites d'exécution et des conditions d'exécution imprévues

4.5. Usages

Voici quelques règles en usage dans l'écriture des requêtes permettant d'en optimiser l'exécution :

MAUVAIS	BON	Pourquoi?
Sélectionnez `SELECT * ...`	Sélectionnez `SELECT col1, col2 ...`	Le SGBDR doit faire un effort important pour rechercher les colonnes adéquates.
Sélectionnez `... col>='val1' AND col=<'val2' ...`	Sélectionnez `... col BETWEEN 'val1' AND 'val2' ...`	L'opérateur BETWEEN est optimisé (sinon à quoi servirait-il ?)
Sélectionnez `COUNT(col) ...`	Sélectionnez `COUNT(*) ...`	Préférez le COUNT(*), le moteur va piocher dans les statistiques, coût voisin de zéro !

Sélectionnez
`CASE col WHEN ... THEN ... ELSE ... END`
	Sélectionnez
	`COALESCE(...)`
	ou
	`UNION`
		Remplacer les CASE par des COALESCE ou des opérations ensemblistes de type UNION, chaque fois que cela est possible, car la structure CASE est d'un coût exorbitant.

Sélectionnez `SELECT DISTINCT ...`	Sélectionnez `SELECT ...`	Évitez le mot clef DISTINCT lorsque cela n'est pas d'une absolue nécessité. Le distinct oblige à dédoublonner donc trier et si les résultats sont unique c'est du temps de perdu.
Sélectionnez `... IN (1, 2, 3) ...`	Sélectionnez `... BETWEEN 1 AND 3 ...`	Évitez le IN lorsque le BETWEEN suffit
		Simplifiez les expressions en ayant si possible une seule

Sélectionnez `... WHERE col1+1 = col2+5 ...`	Sélectionnez `... WHERE col1 = col2 + 4`	colonne indexée de part ou d'autre des opérateurs de comparaison, afin d'activer les index.
Sélectionnez `... WHERE EXISTS (SELECT Col1, Col2 ...`	Sélectionnez `... WHERE EXISTS (SELECT * ...`	Dans ce cas (requête imbriquée avec opérateurs EXISTS) l'optimiseur remplace le caractère * est remplacé par une constante appropriée
Sélectionnez `... WHERE EXISTS (SELECT ...`	Sélectionnez `... WHERE ... IN (SELECT ...`	Lorsque cela s'avère possible, remplacez l'opérateur [NOT] EXISTS par un opérateur [NOT] IN.
Sélectionnez `... WHERE col1 IN (SELECT col1 ...`	Sélectionnez `...FROM table1 t1 INNER JOIN table2 t2 ON t1.col1 = t2.col1 ...`	Lorsque cela est possible remplacer les sous requêtes avec opérateur IN par des jointures.
Sélectionnez `VARCHAR`	Sélectionnez `CHAR`	Préférez le CHAR Lorsque la colonne de la table est sollicité en recherche et/ou que l'on y ajoute un index.
Sélectionnez `NCHAR, NVARCHAR`	Sélectionnez `CHAR, VARCHAR`	Préférez le CHAR/VARCHAR au NCHAR/NVARCHAR lorsque l'application n'est pas multilange. Le coût de stockage est divisé par deux.
Sélectionnez `FLOAT`	Sélectionnez `DECIMAL`	Pour les calculs financiers qui ne doivent pas générer d'erreurs d'écarts d'arrondis.

4.6. Documentation

On veillera à implanter dans la base de données, une table paermettant de décrire les objets de la base.

Une telle table, de nom TS_DESCRIPTION_DSC pourra prendre la forme suivante :

TS_OBJ_NAME_DSC	Nom de l'objet
TS_OBJ_TYPE_DSC	Nature de l'objet (table, vue, procédure, fonction, trigger...)
TS_ATB_NAME_DSC	Nom de l'attribut
TS_ATB_TYPE_DSC	Type d'attribut (colonne de table, paramètre de procédure ou de fonction)
TS_ATB_ORDER_DSC	Position ordinale de l'attribut
TS_ATB_LENGTH_DSC	Longueur de l'attribut
TS_ATB_REQUIRED_DSC	Obligatoire
TS_DESCRIPTION_DSC	Description de l'objet (à destination des utilisateurs)
TS_OBSERVATION_DSC	Annotation de l'objet (à destination des développeurs et administrateurs)
TS_ACCESS_RULE_DSC	Règle d'accès (par exemple, combinaison binaire pour : 1 : utilisateurs, 2 : administrateurs, 4 : développeurs, 8 : chefs de projet ...)
TS_APPLICATION_DSC	Règle d'utilisation par les applications clientes (par exemple, combinaison binaire pour : 1 : paye, 2 : comptabilité, 4 : gestion commerciale, 8 : ...)