Collection de logiciels libres pour l'enseignement et le développement

CoSQL - environnement SQL commun du CoLOED

Présentation générale

CoSQL_SCL

Luc Lavoie *(luc.lavoie@usherbrooke.ca)*

—

CoLOED/SQL/CoSQL_SCL, version 1.0.3a, en date du 2024-08-12

Sommaire

CoSQL est l'environnement SQL commun du CoLOED. Il comprrend les sous-composants suivantes :

- "Base" : un ensemble de types de base prédéfinis, les opérateurs logique d'implication et d'équivalence et un opérateur d'égalité relationnelle.
- "Verif" : des routines de gestion des assertions.
- "IMEX" : des routines d'import-export de fichiers au droit du serveur.
- "Param" : des routines de gestion de paramètres internes propres à la BD.

Mise en garde

Le présent document est en cours de révision ; en conséquence, il est incomplet et peut contenir des erreurs.

Historique

diffusion	resp.	description
2024-08-12	LL	Intégration initiale de la documentation.

Table des matières

Introduction

CoSQL est l'environnement SQL commun du CoLOED.

Les parties publiques et privées n'ont pas été distinguées, puisqu'*a priori* tous les composants de la BD l'incorporant peuvent faire appel à ses entités, mais seules les personnes responsables de la BD devraient être en mesure de les mettre à niveau.

Les sous-composants disponibles sont :

- "Base" [sans état, sans introspection] :
  - Types génériques : Text, "Entier", "Cardinal", "Ordinal", "Estampille".
  - Opérateurs logiques : "bool_imp", "bool_eqv".
- "Base", extension « rel » [sans état, **avec** introspection] :
  - Opérateur d'égalité relationnelle : "rel_ega".
- "Verif" [sans état, sans introspection ; dep("Base")] :
  - Routines de gestion des assertions : "Asserter", "Asserter_exception", "Manifester", "Manisfester_exception".
- "IMEX" [sans état, sans introspection ; dep("Base")] :
  - Routines d'import-export de fichiers au droit du serveur.
  - Pour le moment limité aux formats CSV (le délimiteur de champ pouvant cependant être paramétré).
- "Param" [**avec** état, sans introspection, stable ; dep("Base")] :
  - Gestion de paramètres internes propres à la BD.
- "IMEX", extension « par » [sans état propre, sans introspection, stable ; dep("IMEX", "Param", "Base")] :
  - Extension de "IMEX" relativement à un dépôt (répertoire) associé à la BD par l'entremise de "Param".
  - Typiquement, ce dépôt est localisé sur le serveur hébergeant le SGBD et le compte propriétaire du service associé au SGBD doit y avoir accès.

CoSQL est le fruit de la fusion et de la refonte de GRIIS_Base et Metis_Base.

Notes de mise en oeuvre

Le code ne doit pas être soumis à un formateur, car la mise en forme opérée par ce dernier introduit généralement des erreurs, notamment :

- guillemets fautifs autour du mot réservé `value` ;
- guillemets fautifs autour de l'opérateur `||`.

De plus, les formateurs « cochonnnent » généralement la présentation, notamment :

- ils analysent incorrectement les corps de routines SQL (begin atomic … end) ;
- la gestion des fins de ligne et des indentations n'est pas uniforme.

Bien que la version 12 de PostgreSQL soit encore soutenue (au 2024-07-31), il a été décidé de ne démarrer le soutien de CoLOED qu'à partir de la version 14 pour les raisons suivantes :

- le traitement des déclarations de routines conformes au standard ISO 9075:2016 a été offert dès la version 11.4, de multiples erreurs (tant du compilateur, que du connecteur JDBC que d'autres outils connexes) sont demeurées présentes jusqu'à la version 14.
- le soutien de la version 12 cessant le 2024-11-14, celui de la version 13, le 2025-11-13 (voir https://www.postgresql.org/support/versioning/), il n'est pas apparu justifié de consacrer des efforts au développement et à la mise en place des contournements nécessaires puisque nous ne prévoyons diffuser publiquement CoLOED qu'à partir de la mi-novembre 2024.

Contributeurs

```
(CK01) Christina.Khnaisser@USherbrooke.ca
(LL01) Luc.Lavoie@USherbrooke.ca
(MD01) Marc.Dupuis@USherbrooke.ca
(SD01) Samuel.Dussault@USherbrooke.ca
```

Adresses, droits d'auteur et copyright

```
2016-2019
Groupe Μητις (Metis)
Faculté des sciences
Université de Sherbrooke (Québec) J1K 2R1
CANADA
http://info.usherbrooke.ca/llavoie/
```

```
2020-2023
GRIIS (Groupe de recherche interdisciplinaire en informatique de la santé)
Faculté de médecine et sciences de la santé et Faculté des sciences
Université de Sherbrooke (Québec) J1K 2R1
CANADA
https://griis.ca/
```

```
2024-...
CoFELI (Collectif francophone pour l'enseignement libre de l'informatique)
https://github.com/CoFELI
```

Licences

```
Code sous licence LILIQ-R+ (https://forge.gouv.qc.ca/licence/liliq-v1-1/).
Documentation sous licence CC-BY-4.0, (https://creativecommons.org/licenses/by/4.0/).
```

Tâches projetées

```
2022-10-03 (LL01) : Faciliter la variation de la politique Asserter/Manifester (PAM).
  * Ajout d'une série de routines ayant un paramètre supplémentaire déterminant
    la PAM désirée.
  * Ajout d'une série de routines se conformant à une PAM établie globalement.
```

Tâches réalisées

```
2024-08-12 (LL01) : Documentation de la version 103.
2024-07-24 (LL01) : Harmonisation pour Μητις, GRIIS et CoFELI.
2022-09-28 (LL01) : Routines de vérification de l'occurrence d'exceptions.
  * Ajouts :
      _assert_exception, Assert_Exception,
      _manifest_exception, Manifest_Exception.
  * Refactorisations mineures.
  * Commentaires.
2022-07-25 (LL01) : Recentrage du module
  * Retrait de Taux et Pourcentage.
  * Déplacement et renommage des entités relatives aux courriels dans IETF5222.
2022-06-24 (LL01) : Déplacement de la gestion des codes vers le module Code.
```

```
2022-06-19 (LL01) : Intégration des propositions Μητις et GRIIS.
2020-03-21 (LL01) : Création initiale pour le projet ProtoGEM.
2016-01-03 (LL01) : Création initiale pour le groupe Μητις.
```

Références

```
[std] https://github.com/CoLOED/Scriptorum/blob/main/pub/STD-SQL-01_NT.pdf
```

1. Base

1.1. Types textuels

Plusieurs dialectes SQL propose un type textuel non borné a priori. Celui-ci est le plus souvent dénoté « Text » (TEXT, Text ou text). Ce type est présumé pour être manipulable par tous les opérateurs utilisant un opérande textuel (CHAR ou VARCHAR) prévus par la norme ISO 9075:2016.

Si le dialecte utilisé ne définit pas un tel type, "CoSQL" une définition la moins fautive possible. Dans ce contexte, l'identifiant ne doit donc pas être délimité.

1.2. Types entiers

Définition des types entiers les plus étendus associés aux opérandes des opérateurs pris en compte par la norme ISO 9075:2016 : "Entier", "Cardinal", "Ordinal". Afin de faciliter le diagnostic, la coercition « not null » leur est appliquée afin de provoquer la levée d'une exception le plus précocement possible en cas d'indétermination.

Pour chacun de ces trois types e parmi {entier, cardinal, ordinal}, les fonctions "e_min"() et "e_max"() retournent les valeurs minimals et maximales.

1.3. Type Booléen

L'implication et l'équivalence sont des opérateurs logiques usuels, mais non pris en compte par la norme ISO 9075:2016. Les opérateurs sont définis par "CoSQL".

- "bool_imp" (a Boolean, b Boolean)
- "bool_eqv" (a Boolean, b Boolean)

1.4. Types temporels (Estampille)

Le type Timestamp est malheureusement très variablement mis en oeuvre par les dialectes SQL. De façon à uniformiser les vérifications et les traitements, le type "Estampille" est défini comme un point temporel (donc discret et fini) compris entre "estampille_min" et "estampille_max". La granularité est est établie par la durée du chronon ("estampille_chronon", de type « Interval ») et le nombre de chronons par seconde ("estampille_ncs", de type "Ordinal") sous la contrainte suivante :

```
"estampille_chronon" * "estampille_ncs" = 1 seconde
```

Par ailleurs, l'échelle de référence du type "Estampille" est le temps universel coordonné (UTC) au méridien de Greenwich (0000). Toute valeur indexée en fonction d'un fuseau horaire horaire doit être convertie (elle l'est implicitement en PostgreSQL en regard du type « Timestamp », du moins depuis la version 12).

Finalement sont définies des fonctions "gen_VT", "gen_TT" et "gen_TC" permettant d'obtenir :

- le temps de validité (*valid time*, *user time*) ;
- le temps de transaction (*transaction time*, *system time*, *log time*) ;
- le temps courant (*current time*, *real time*).

Mise en garde

« In a literal that has been determined to be timestamp without time zone, PostgreSQL will silently ignore any time zone indication. That is, the resulting value is derived from the date/time fields in the input value, and is not adjusted for time zone.

For timestamp with time zone, the internally stored value is always in UTC (Universal Coordinated Time, traditionally known as Greenwich Mean Time, GMT). An input value that has an explicit time zone specified is converted to UTC using the appropriate offset for that time zone. If no time zone is stated in the input string, then it is assumed to be in the time zone indicated by the system's TimeZone parameter, and is converted to UTC using the offset for the timezone zone.

When a timestamp with time zone value is output, it is always converted from UTC to the current timezone zone, and displayed as local time in that zone. To see the time in another time zone, either change timezone or use the **AT TIME ZONE** construct (see Section 9.9.4).

Conversions between timestamp without time zone and timestamp with time zone normally assume that the timestamp without time zone value should be taken or given as timezone local time. A different time zone can be specified for the conversion using AT TIME ZONE. »

Attendu que les fonctions du système (CURRENT_TIMESTAMP, CLOCK_TIMESTAMP, LOCAL_TIMESTAMP, etc.) sont toutes de type TIMESTAMP WITH TIME ZONE, les conversions implicites sont inéluctables. Il est donc TRÈS difficile de faire des traitements cohérents à moins d'imposer au système le fuseau UTC et de s'assurer qu'aucune commande (telle SET TIME ZONE) ne vient compromettre cet état.

Pour cette raison, un futur sous-composant CoSQL_Tempus proposera une représentation adéquate du temps (via le type TimePoint et ses dérivés) en le distinguant des diverses conventions calendaires et en l'affranchissant des fuseaux horaires.

Références

- https://www.bipm.org/fr/
- https://www.postgresql.org/docs/current/datatype-datetime.html

Sous PostgreSQL, la représentation et le calcul utilisent le type float8 (numérique flottant) ce qui entrainte une imprécision relative et des erreurs d'arrondis. Pour une représentation discrète, voir le type "Timepoint" du sous-composant CoSQL_Tempus ou du module "UHF".

1.5. Relations

La fonction "rel_ega" est un palliatif à l'absence de l'opérateur d'égalité entre deux relations (tables) en SQL. Chaque table à comparer y est désignée par une valeur de type Text représentant la dénotation complète de la table comme prescrit par la norme ISO 9075:2016 (préfixation l'identifiant du schéma ou non, identifiants simples ou délimités, etc.).

"rel_ega" (z_t1 Text, z_t2 Text) : Boolean;

- Soit deux tables de même type, la fonction détermine si elles sont égales au sens relationnel du terme.
- Deux tables sont du même type si elles partagent exactement les mêmes attributs (mêmes identifiants des mêmes types).
- Deux tables sont égales au sens relationnel si elles sont composées exactement des mêmes tuples.
- Rappel : les tables représentant ici des relations sont des ensembles. Elles ne peuvent donc pas comprendre de tuples en double.

Note de mise en oeuvre

- La définition de "rel_ega" a été séparée des autres définitions du sous-composant CoSQL_Base, car sa mise en oeuvre fait appel à la commande EXECUTE de SQL. Dans plusieurs contextes (pour des raisons de sécurité, de vérifiabilité, etc.), le recours à cette commande est proscrit. En séparant cette définition des autres, il est plus facile d'inclure ces dernières sans déroger aux prescriptions.

TOIMPROVE 2024-08-06 (LL01) Optimisation possible de "rel_ega" ?

- Il serait approprié de déterminer si la mise en oeuvre par double différence est plus efficiente celle par

jointure présentement utilisée.

2. Verif

2.1. Gestion des tests et des assertions

Les routines de ce sous-composant ont pour but de faciliter et d'uniformiser les tests des composants.

"test_identification" (test Text, partie Text) : Text;

Dénotation uniforme horodatée (UTC) et textuellement affichable d'un test (ou d'une partie d'un test)

"Asserter" (message Text, predicat Boolean);

- Si le prédicat est vrai, ne fait rien.
- Si le prédicat est faux, signale une exception comprenant le message préfixé par ERR.

"test" (message Text, predicat Boolean) : Boolean;

- Si le prédicat est vrai, signale une note (NOTICE) comprenant le message préfixé par POS.
- Si le prédicat est faux, signale un avertissement (WARNING) comprenant le message préfixé par NEG.
- Retourne la valeur du prédicat.

"Manifester" (message Text, predicat Boolean);

- Si le prédicat est vrai, signale une note (NOTICE) comprenant le message préfixé par POS.
- Si le prédicat est faux, signale un avertissement (WARNING) comprenant le message préfixé par NEG.

"Asserter_exception" (message Text, instruction Text);

- Si l'instruction lève une exception, la neutralise et ne fait rien d'autre.
- Si l'instruction ne lève pas d'exception, signale une exception comprenant le message préfixé par ERR.

"test_exception" (message Text, instruction Text) : Boolean;

- Si l'instruction lève une exception, la neutralise, signale une note (NOTICE) comprenant le message préfixé par POS puis retourne vrai.
- Si l'instruction ne lève pas d'exception, signale un avertissement (WARNING) comprenant le message préfixé par NEG puis retourne faux.

"Manifester_exception" (message Text, instruction Text) : Boolean

- Si l'instruction lève une exception, la neutralise puis signale une note (NOTICE) comprenant le message préfixé par POS.
- Si l'instruction ne lève pas d'exception, signale un avertissement (WARNING) comprenant le message préfixé par NEG.

2.2. Éléments obsolètes

Le dialecte PostgreSQL antérieur à la version 11.4 n'offrait pas la possibilité de définir des procédures conformément aux prescriptions de la norme ISO 9075:2016. En lieu et place, une fonction pouvait être définie avec un pseudo-type « void » et appelée par le biais de l'instruction « select ».

Ce dernier mécanisme est devenu redondant à partir de la version 11.4 qui mit à disposition le mécanisme prévu par la norme (avec certaines erreurs, il est vrai, qui ne seront corrigées qu'à partir de la version 14). Depuis ce moment, les deux formes de procédures ont été offertes par de nombreux modules de CoLOED (ou de ses ancêtres).

Les versions de PostgreSQL antérieures à 12 étant désormais caduques (non « supportées »), les formes obsolètes ne sont plus nécessaires et seront définitivement retirées au 2026-01-01. Entretemps, afin de faciliter la transition des corpus de code non encore migrés, le présent fichier permet de définir les formes

désormais obsolètes.

- DEPRECATED 2024-07-25 (LL01) : void function "proc_assert"
- DEPRECATED 2024-07-25 (LL01) : void function "proc_manifest"
- DEPRECATED 2024-07-25 (LL01) : void function "proc_assert_exception"
- DEPRECATED 2024-07-25 (LL01) : void function "proc_manifest_exception"

3. Param

Les routines du sous-composant CoSQL_Param permettent de gérer des paramètres internes de la base de données dans laquelle le schéma "CoSQL" est défini.

Un paramètre est défini par trois valeurs de type Text : un identifiant, une valeur et une annotation (considérée comme un commentaire explicatif quant à sa nature, son rôle et sa portée).

Un paramètre peut être retiré (indéfini) sur la base de son identifiant. On ne peut redéfinir un paramètre existant sans le retirer au préalable. La valeur d'un paramètre défini peut être obtenue, modifiée ou affectée à un autre paramètre déjà défini. L'annotation d'un paramètre ne peut être modifiée. Il faut retirer le paramètre et le redéfinir.

"Parametre_interne_definir" (z_parametre Text, z_valeur Text, z_annotation Text);

- Définir le paramètre, puis lui associer la valeur et l'annotation.

"Parametre_interne_retirer" (z_parametre Text);

- Retirer le paramètre.

"Parametre_interne_modifier" (z_parametre Text, z_valeur Text);

- Affecter la valeur au paramètre.

"Parametre_interne_copier" (z_source Text, z_destination Text);

- Affecter la valeur du paramètre source au paramètre destination.

"Parametre_interne_valeur" (z_source Text);

- Retourner la valeur du paramètre.

4. IMEX

4.1. Interfacte directe

Module TRÈS SIMPLE d'importation et d'exportation de tables à partir de fichiers présents dans l'environnement du SGBD et présumés accessibles à ce dernier.

Notes

Les fichiers importés et exportés sont présumés être au format .csv bien que le délimiteur soit paramétrable. Le suffixe du nom de fichier n'est pas imposé pour autant et doit conséquemment faire partie du nom. Une généralisation pourrait être souhaitable.

"Importer" (z_nom_table Text, z_repertoire Text, z_fichier Text, z_delimiteur char default ',');

- Les donnnées du fichier z_fichier localisé dans le répertoire z_repertoire seront versées dans la table _nom_table (au besoin le nom doit inclure le nom du schéma).
- Les données elles-mêmes y sont délimitées par le le symbole z_delimiteur (à défaut, la virgule).
- Le répertoire depuis lequel le fichier est importé est une chaine caractères en accord avec l'environnement d'exécution.

"Exporter" (z_requete Text, z_repertoire Text, z_fichier Text, z_delimiteur char default ',');

- Les donnnées produite par la requête z_requête seront versées dans le fichier z_fichier localisé dans le

répertoire z_repertoire.

- Les données elles-mêmes y seront délimitées par le le symbole z_delimiteur (à défaut, la virgule).
- Le répertoire depuis lequel le fichier est exporté est une chaine caractères en accord avec l'environnement d'exécution.

4.2. Interface paramétrée

Sous-composant offrant l'importation de fichiers [et l'exportation de tables] à partir [au droit] de l'environnement du SGBD depuis le répertoire désigné par le paramètre interne "parametre_importation" () ["parametre_exportation" ()].

Notes

- Les fichiers importés et exportés sont présumés être au format .csv bien que le délimiteur soit paramétrable. Le suffixe du nom de fichier n'est pas imposé pour autant et doit conséquemment faire partie du nom.
- Les droits d'accès sont présumés gérés par ailleurs.
- Une généralisation pourrait être souhaitable.

"Importer_par" (z_requete Text, z_fichier Text, z_delimiteur char default ',');

- Même fonctionnalité que "Importer" si ce n'est que le répertoire est celui déterminé globalement par le paramètre interne "parametre_importation" ().

"Exporter_par" (z_requete Text, z_fichier Text, z_delimiteur char default ',');

- Même fonctionnalité que "Exporter" si ce n'est que le répertoire est celui déterminé globalement par le paramètre interne "parametre_exportation" ().

Conclusion

Une consultation doit être entreprise avant la pulication du composant afin de déterminer son adéquation et les prolongements souhaitables.

Plus particulièrement, les aspects suivants doivent être approuvés :

- l’utilisation des identificateurs délimités ;
- le choix des dénomination des routines du sous-composant "Param" (non conforme aux resommandatios de STD-ENV-01) ;
- la forme et le contenu de la documentation.

A. Correctifs récents

La présente annexe comprend la liste des correctifs récemment appliqués.

FIXED 2023-07-27 (LL01) Test Estampille #2 est hors borne sous psql, mais pas sous Datagrip

- Selon la documentation, il ne devrait pas y avoir d'erreur.
- Voir https://www.postgresql.org/docs/16/datatype-datetime.html (inchangé de v12 à v16)
- CORRECTION
  - À cause du décalage horaire implicite lors des conversions implicites entre Timestamp et Timestamp with time zone, il faut toujours opérer une conversion explicite en utilisant l'opérateur cast. C'est désormais le cas.

FIXED 2023-07-27 (LL01) Test Estampille #3 concluant sous psql, mais pas sous DataGrip.

- L'expression `"estampille_min"() - "estampille_chronon"()` doit lever une exception.
- CORRECTION
  - À cause du -infinity, la détection échoue tant que la valeur n'est pas affectée à une variable de type Estampille. Elle ne le sera pas si elle est affectée à une variable de type Timestamp (indépendamment de la présence ou non du time zone).
  - Le test a donc été modifié afin d'évaluer l'expression suivante : `cast ("estampille_min"() - "estampille_chronon"() to "Estampille")`

Produit le 2024-08-13 19:24:26 UTC

Collection de logiciels libres pour l'enseignement et le développement