Évènements et Opérations dans le Réseau Sémantique Universel

Après une présentation générale (voir La Lettre n^{os}26 à 28) et quelques commentaires complémentaires (voir La Lettre n°30), voici quelques nouvelles réflexions, partiellement inspirées par l'article sur le modèle OOE[1] (que vous pouvez lire par ailleurs dans cette même Lettre).
Sommes-nous partis pour une chronique trimestrielle ? Pourquoi pas ? (surtout si un véritable débat - ou mieux, une commission ADELI[2] - s'ouvre).

Un nom !

Quelques Adéliens m'ont fait remarquer que le nom de "Réseau Sémantique Universel" (ainsi que son acronyme R.S.U.) n'était peut-être pas très "commercial" et pouvait rebuter certaines personnes. Alors, qui aurait une idée pour donner un meilleur nom ?

Personnellement, j'oserais bien baptiser cet ensemble… MERISE/3. Après tout, il faut savoir que :
a) L'AFCET n'existe plus, et le groupe qui faisait évoluer MERISE s'est éparpillé dans la nature. Il serait bien que le flambeau soit repris.
b) L'appellation MERISE/2 n'a jamais été le nom officiel des évolutions apportées par l'AFCET à MERISE, mais bien celui d'une version spécifique et prorpiétaire de SEMA-METRA. MERISE/3 serait beaucoup plus proche d'une évolution des spécifications de l'AFCET que de l'usine à gaz[3] qu'est MERISE/2.
c) Il est temps de montrer que MERISE, loin d'être une méthode dépassée, était en fait dès sa création en avance (après tout, ne contient-elle pas les notions d'objet et d'événement… depuis 1980 !), et que si elle s'était un peu endormie sur ses lauriers, elle est toujours parfaitement apte à intégrer les concepts les plus récents, comme je l'ai démontré dans les articles précédents.
d) MERISE est une marque déposée (par le Centre d'Études des Systèmes d'information des Administrations le 17/11/1978, renouvelée le 9/12/1988), ainsi que MERISE OBJET (par X. Castellani, le 10/12/1991).

Le débat est ouvert…

RSU contre OOE

Contre, oui… mais tout contre, en fait.

En effet, si on compare les base de OOE avec celles du RSU, on y trouve un très fort parallélisme, que présente le tableau ci-dessous (voir diagrammes de l'article sur OOE et le métamodèle du Réseau Sémantique présenté dans LA LETTRE n°27). Pour l'essentiel, les divergences viennent du fait que le modèle RSU est beaucoup plus complet que le modèle OOE présenté dans ces pages.

---

[1] *C'est un des privilège des membres du Comité de lecture : connaître les articles avant leur parution. J'en profite honteusement (et sans le moindre scrupule) ! Mais honnêtement quand même : les auteurs de l'article cité ont également connaissance de celui-ci pour approbation préalable…*

[2] *C'est un appel aux amateurs…*

[3] *C'est bien sûr une opinion personnelle (et je la partage). MERISE contient 6 aspects ; MERISE/2 en définit 12 ! MERISE 3 n'en a probablement besoin que de 3…*

Métamodèle OOE	Métamodèle RSU	Commentaires
objet OBJET	objet OBJET	
objet ÉVÉNEMENT *trigger*	objet PRÉDICAT	
objet OPÉRATION	objet ACTION	
relation ÉVÉNEMENT constate_changement _d'état_de OBJET	action Prédicat.Modifier()	voir aussi les relations PRÉDICAT déduit_de ATTRIBUT et PRÉDICAT vérifié_par VALEUR
relation ÉVÉNEMENT déclenche_l'exécution _de OPÉRATION	relation PRÉDICAT déclenche RÈGLE	Rappel : RÈGLE est un sous-type de ACTION
relation OPÉRATION modifie_l'état_de OBJET	relation ACTION modifie ATTRIBUT	Rappel : on a toujours ATTRIBUT appartient_à OBJET|ASSOCIATION|TYPE|ACTION[4]
message	objet TÂCHE	

Un petit reproche que je ferai à Mesdames Foucaut et Thiéry, c'est de limiter leur approche au modèle relationnel qui, s'il est de loin le plus répandu actuellement - car le plus simple de conception - exclut de prime abord l'approche objet[5], toutes les adaptations faites dans ce sens n'étant que des "verrues". Pour moi, seul un vrai SGBD Objet permet une véritable implémentation des notions d'événement et opération.

Le débat est ouvert…

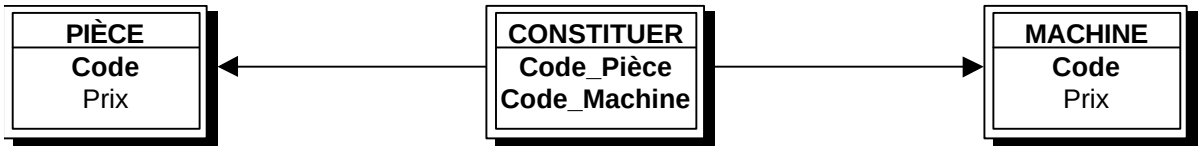
Les requêtes : comparatif entre SQL et RSU

Pour apporter de l'eau au moulin du RSU, définissons une toute petite application, et voyons comment définir une requête dans chacun des modèles relationnel et RSU.

Le MCD de l'application

Il s'agira tout simplement de définir des machines constituées de pièces, sachant qu'une machine est généralement composée de plusieurs pièces, et qu'une pièce peut servir dans plusieurs machines (pour rester simple, on passera sur le nombre de pièces identiques dans une même machine). Pièce et machine ont comme attributs un code (identifiant) et un prix. Nous obtenons donc le MCD suivant :

Dans un SGBD relationnel, on obtient le groupe de trois tables ci-dessous (les flèches indiquent les contraintes d'intégrité référentielle) :



Dans un SGBD objet, le MCD correspond à l'implémentation physique.

La requête

Nous allons maintenant demander pour chaque pièce la liste des machines dont elle constitue plus de 10% du prix total.

---

[4] *Le séparateur | signifie : ou exclusif.*

[5] *Les contraintes d'intégrité n'étant qu'un pis-aller pour prendre en compte les cardinalités (0,1) ou (1,1).*

Requête relationnelle (SQL)

Pour faciliter la compréhension des lecteurs non habitués - ou ne connaissant pas - SQL, nous allons franciser la syntaxe. On travaille en deux temps :

a) Définition de la requête :

```
DÉFINIR PièceChère CURSEUR POUR
    LISTER Pièce.Code,
           Pièce.Prix,
           Machine.Code,
           Machine.Prix
    DE Pièce, Machine, Constituer
    DONT Pièce.Code = Constituer.Code_Pièce
      ET Machine.Code = Constituer.Code_Machine
      ET Pièce.Prix > Machine.Prix * 0,1
    TRIÉ_PAR Pièce.Code, Machine.Code ;
```

Explication : on demande de récupérer quatre colonnes de deux tables, en effectuant une jointure conditionnelle avec une troisième table[6], et en filtrant les données récupérées avec une condition. Le système, à l'ouverture de la requête, lira donc les trois tables, effectuera toutes les combinaisons possibles respectant les conditions précitées, chargera le tout dans une table temporaire (en mémoire… virtuelle) qui sera enfin triée selon les critères spécifiés.

b) Exécution :

- OUVRIR PièceChère déclenchera la constitution de la liste ;
- chaque appel de LIRE PièceChère fournira un élément de la liste ;
- FERMER PièceChère fermera et détruira la requête.

Sur des tables importantes (plusieurs millions de lignes), la seule exécution de OUVRIR peut durer très longtemps (j'ai examiné certaines requêtes DB/2[7] prenant 20 minutes de temps-machine sur un IBM ES9000 quadriprocesseur, soit 1,4% de sa capacité quotidienne totale ! Et je n'ai pu faire mieux que de les ramener à… 16 minutes !). De plus, ce temps est consommé chaque fois que la requête est effectuée, car le système doit reconstituer la liste.

Et si on veut d'autres colonnes, ou un ordre différent, ou un filtrage sur certaines pièces ou machines, ce sont autant de requêtes différentes.

Bref, ce n'est pas le pied…

Bien sûr, avec certains SGBD relationnels très évolués, il est possible de définir une table de requêtes et des "triggers" qui permettent de ne pas tout reconstruire à chaque fois, mais on atteint une certaine complexité…

Requête Objet

Nota : la syntaxe utilisée ici n'est pas "standard" des SGBD Objet – c'est celle du système EPHIBASE.

On procède également en deux temps :

a) Création de la requête :

```
CRÉER_ENTITÉ Requête PièceChère
   Sélection='(Pièce est_partie_de Machine) DONT Pièce_Prix > (Machine.Prix * 0,1)'
   Clé=[Pièce.Code Machine.Code]  Permanent ;
```

---

[6] En effet, si on peut définir dans le SGBD que le contenu de Contituer.Code-Pièce doit correspondre à une valeur existante de Pièce.Code, il n'y a aucun automatisme avec SQL, et on doit tout détailler.

[7] Je sais, DB/2 est loin d'être le meilleur… mais c'est le plus répandu sur les gros systèmes IBM. Et de toute façon, c'est pas moi qui ai choisi !

Explication : Le système va créer une occurrence d'objet, de classe Requête, dont l'identifiant est PièceChère, avec 2 attributs. Les actions associées à cette classe et à ses attributs (lors de la définition de la classe Requête) vont se déclencher et :

- Parcourir les occurrences de la relation Pièce est_partie_de Machine. Celles qui répondront à la condition seront reliées à l'objet PièceChère par une relation Requête concerne (Pièce est_partie_de Machine), dans la clé de laquelle seront placées les valeurs spécifiées par l'attribut-liste Clé. L'attribut Permanent (booléen, donc 'Permanent' équivaut à 'Permanent=oui' et 'non Permanent' équivaut à 'Permanent=non') indiquant que c'est une requête devant resservir (dans le cas contraire, l'objet - avec ses relations - serait détruit après un certain délai).
- Chaque fois qu'une mise à jour touchant l'un ou l'autre des objets, attributs et relations citées dans la requête, une mise à jour de celle-ci aura lieu. Elle sera donc toujours "up-to-date".

À noter que la syntaxe de création d'une requête est ni plus ni moins que la syntaxe standard qui permet de créer n'importe quel objet.

b) Exécution de la requête :

```
LISTER_RELATIONS Requête PièceChère concerne *
   Pièce.[Code Prix] Machine.[Code Prix]
```

Explication : il s'agit d'une demande de liste standard, sachant qu'on demande ici de lister, pour l'objet de classe Requête et d'identifiant PièceChère, les relations de type concerne, pour toutes entités reliées (qu'elles soient objets ou relations), et qu'on veut récupérer deux attributs de Pièce et deux de Machine.

Les principales différences avec SQL sont que :

- la requête n'a été crée qu'une fois (même si cette création prend autant, ou même un peu plus, de temps qu'en relationnel, c'est moins important, puisque non répétitif). Et le temps consommé pour la maintenir à jour est faible (probablement 1% ou moins du coût de création, sur des grosses classes) ;
- on peut à chaque demande spécifier une liste d'attributs de son choix : on n'est donc pas figé comme avec SQL ;
- il est possible, dans la demande de la liste, de demander un filtrage supplémentaire (par exemple DONT Machine.Prix > 10000) : on n'est pas figé ici non plus.

Conclusion

Il est vrai que je n'ai pas lu l'ouvrage de Mmes Foucaut et Thiéry, mais je le ferai si cela s'avère nécessaire, car il est probable qu'un comparatif plus complet entre les deux approches est susceptible d'enrichir l'une comme l'autre, et de donner lieu à une synthèse intéressante.

Quand aux conclusions proprement dites, je laisse au lecteur le soin d'en tirer celles qui lui conviennent… et de m'en faire part si possible. La création d'une commission sur ce sujet serait un grand pas en avant (non non, nous ne sommes pas au bord du gouffre !).

À la prochaine...▲

Jean-Luc Blary
e-mail : jlblary@nordnet.fr