Groupe de recherche en modélisation et gestion de données

Décomposition relationnelle avec intervalles ouverts sur leurs deux bornes à la fois

METIS_UHF_NT-2023-06
Christina Khnaisser *(christina.khnaisser@usherbrooke.ca)*
Luc Lavoie *(luc.lavoie@usherbrooke.ca)*
(les auteurs sont cités en ordre alphabétique nominal)

—

Metis/uhf-base/UHF_NT, version 1.2.0c, en date du 2025-02-25

— document de travail, ne pas citer à l'externe —

Sommaire

¡¡ À réviser !!

Proposition d'une nouvelle décomposition des relations comportant des attributs clés de type intervalle dans le but de réduire les pertes de données découlant des restrictions sur certaines opérations lorsqu'une des bornes de certains opérandes de type intervalle est indéfinie. Plus particulièrement, la classe des opérations entrainant une consolidation entre les partitions @xe, @bx ou @be (¿¿autant une paire d'entre elles que les trois ensemble??). La solution proposée, l'intégration de la partition @df, permet de réduire la perte de données dans un tel cas.

La présente version ne comprend que les sections 1 et 2. Les sections 3 à 6 suivront bientôt.

Mise en garde

La présente note technique est en cours d'élaboration ; en conséquence, elle est incomplète et peut contenir des erreurs.

Historique

diffusion	resp.	description
2025-02-05	LL	Revue en vue de la présentation en IFT723.
2024-02-05	LL	Adaptation aux conventions de CoLOED.
2023-05-19	LL	Corrections de mise en forme.
2023-02-22	LL	Ébauche initiale.

Table des matières

Introduction

¡¡ À rédiger !!

1. Présentation

Attendu une relation représentant un prédicat et un référentiel défini comme un ensemble fini de points totalement ordonnés, la *relativisation* est l'opération appliquée à la relation de façon à limiter la portée du prédicat associé à un sous-ensemble de points du référentiel. En pratique, la relativisation consiste à ajouter un attribut représentant le référentiel aux attributs de la relation, puis à l'ajouter à chacune des clés candidates de la relation. Dans le cas particulier où une interprétation temporelle est associée au référentiel, l'opération de relativisation est souvent dénommée temporalisation.

En soi, la relativisation ne nécessite aucun aménagement de la théorie relationnelle. Néanmoins, en pratique, certains facteurs associés à la relativisation méritent d'être étudiés plus avant :

1. le très grand nombre de points pouvant être mis en cause est susceptible d'avoir une incidence importante sur l'efficience (en espace mémoire, en temps de calcul et donc en énergie), surtout si plusieurs référentiels sont en cause ;
2. l'association d'une sémantique spécifiquement associée à certains référentiels amène à vouloir développer des opérateurs relationnels, des modèles de schémas et des contraintes en facilitant l'expression (en particulier pour les référentiels spatiaux, temporels et spatiaux-temporels) ;
3. par ailleurs, la co-existence de plusieurs référentiels dans un même modèle appelle à une systématisation de leur traitement afin de simplifier l'élaboration des requêtes, une telle systématisation ne peut vraisemblablement s'opérer que sur une base commune, ici désignée comme la solution à «interprétation neutre» (SIN).

Historiquement, le premier facteur a donc entrainé très tôt (¿¿dès les années 1980??) la recherche d'une représentation plus dense des relations relativisées. L'utilisation d'intervalles de points est l'avenue la plus souvent retenue, même s'il reste encore de nombreuses approches concurrentes et de nombreux problèmes d'optimisation des structures d'indexation et des algorithmes de calcul des opérateurs relationnels. Dans un premier temps, une synthèse de cette approche est donc proposée.

Une généralisation des mécanismes développés dans le contexte d'un référentiel unique est ensuite proposée grâce à l'introduction du concept d'agent. Chaque référentiel de relativisation est associé à un agent propre. Au sein d'un prédicat, le référentiel de relativisation d'un agent *ag* est dénoté RDR[*ag*]. Dans le cas où il y a un seul agent, il est souvent implicite. Auquel cas, le référentiel est noté RDR[]. C'est notamment le cas en bi-temporalité classique où le temps de transaction est implicitement associé au SGBD hébergeant la BD et le temps de validité à «l'organisation», prise dans son ensemble et considérée comme un seul agent. Ainsi une proposition «absolue» est-elle d'abord relativisée par rapport à l'organisation puis la proposition résultante par rapport au SGBD.

On aurait toutefois pu considérer plusieurs temps de validité correspondant à des agents différents au sein de cette organisation (par exemple, dans un hôpital, le service d'urgence, le service de suivi clinique post-hospitalisation et le service des archives).

¡¡ À développer !!.

Par ailleurs, en complément des intervalles, un mécanisme d'horizon de calcul est a été proposé par [Loretnzos 1994, Snodgrass 2000, Date 2003, Date 2014]. Il sera intégré, puis étendu.

Finalement, la sémantique des référentiels (particulièrement des référentiels temporels) a remis au gout du jour un vieux problème auquel aucune solution consensuelle n'a encore été trouvée, celui de la modélisation et du traitement des données manquantes (absentes, non encore disponibles, présumées, etc.). En pratique, nous ne considèrerons ici qu'une seule approche à ce problème, la plus générale, à savoir celle de la modélisation relationnelle par décomposition (projection-jointure, restriction-union ou les

deux). Cette solution possède entre autres avantages de permettre de rendre compte des causes du manque, voire de l'inapplicabilité

2. Exemple de base

Soit

- KT {k1, k2, …, kn} et AT {a1, a2, …, am} deux types,
- R une classe (d'entités),
- k un attribut de R de type KT identifiant uniquement toute entité de R
- a un attribut de R de type AT dépendant fonctionnellement de k

une relvar R (¿¿en 5FN??) représentant le prédicat Q(k,a) est définie comme suit :

```
RELVAR R {k KT, a AT}
  KEY {k}
  PREDICATE Q(k,a)
```

Nous allons présenter ci-après la relativisation non dense à l'aide de points, la transposer à l'aide d'intervalles puis la densifier.

Note 1

La clause PREDICATE permet de définir le prédicat associé à la relvar. Inversement, la fonction PREDICATE_OF permet d'obtenir le prédicat associé à une relvar ou induit par une expression relationnelle sur la base des relvar mises en cause.

Note 2

La relativisation intervient à partir d'une relation en 5FN. Il en découle donc qu'aucun attribut n'est annulable (au sens de SQL). Si on doit modéliser la possibilité de l'absence de certaines données, cela doit avoir été traité au préalable par décomposition de PJ, RU ou les deux. De plus, puisque nous limitons notre exemple initial à un seul attribut non clé, la relation est, dans les faits, en 6FN. Nous généraliserons à plusieurs attributs non clés après avoir généralisé la solution à plusieurs référentiels.

Note 3

Le langage dans lequel les prédicats sont exprimés n'est pas imposé. On peut imaginer autant OWL en relation avec une ontologie fondamentale de base telle que BFO ou DOLCE qu'un texte en français ou en grec faisant appel à des définitions aristotéliciennes.

2.1. Relativisation non dense

Un référentiel de relativisation (RDR) est défini sur la base de points de référence le plus souvent décrits par un sous-type ordinal, par exemple :

```
SUBTYPE PT OF integer CONSTRAINT (ptMin ≤ VALUE ≤ ptMax)
```

Aux fins de l'exemple, $ptMin = 0$ et $ptMax = 99$.

En regard d'un référentiel « neutre » et implicite (sans agent explicite), dénoté par RDR[], la relativisation ponctuelle de R, dénotée R@p, est définie comme suit :

```
RELVAR R@p {k KT, a AT, @p PT}
  KEY {k, @p}
  PREDICATE Q(k,a) ∧ RDR[] = @p
```

Si le prédicat Q(k,a) se lit comme suit

> La valeur de la propriété «Q» de (l'entité «R» identifiée par *k*) est *a*.

L'interprétation relativisée s'interprète comme suit

> [La valeur de la propriété «Q» de (l'entité «R» identifiée par *k*) est *a*] au point *@p* du référentiel RDR.

Les parenthèses ont été ajoutées afin d'expliciter la portée de l'identifiant *k*. Les crochets ont été ajoutés afin d'expliciter la portée du point *@p*.

Exemple non neutre

Si le prédicat Q(k,a) se lit comme suit

> La largeur de la route *k* est de *a* mètres.

Attendu un référentiel de relativisation non neutre défini comme la distance, en kilomètres, depuis l'origine convenue de la route, le prédicat de R@p pourrait donc se lire comme suit

> La largeur de la route *k* est de *a* mètres au kilomètre *@p*.

Si le prédicat Q(k,a) se lit plutôt comme suit

> L'entrepôt dispose de *a* exemplaires du produit *k*.

Attendu un référentiel de relativisation non neutre défini comme le temps écoulé, en minutes, depuis une origine convenue (par exemple, le moment de mise en exploitation de l'entrepôt), le prédicat de R@p pourrait donc se lire comme suit

> À l'instant *@p*, l'entrepôt dispose de *a* exemplaires du produit *k*.

Une telle relvar pourrait avoir la valeur suivante :

```
R@p{
  {k1, a1, 4},
  {k1, a1, 5},
  {k1, a1, 6},
  {k1, a2, 7},
  {k2, a2, 4},
  {k3, a1, 4},
  {k3, a1, 5},
  {k3, a3, 6},
  {k3, a1, 7}
}
```

Note 1

La clause SUBTYPE permet de définir un sous-type d'un type par sa contrainte. Nous réservons la clause

BASETYPE à la définition de type de base (type racine).

Note 2

Intentionnellement, nous avons choisi deux exemples de prédicats où l'ordre des variables n'est pas le même dans les formulations retenues. Cet exemple illustre l'importance du choix de la dénotation nominale plutôt que positionnelle au sein du modèle relationnel. Ce n'est pas le seul exemple. La préservation de la commutativité de la jointure en est un autre.

Note 3

Les fonctions caractéristiques %min et %max permettent de référer aux valeurs limites d'un type ordinal, ainsi

$PT\%min() = ptMin = 0$
$PT\%max() = ptMax = 99$

Note 4

L'expression des référentiels pourrait être différente, ainsi, plutôt que d'utiliser la distance à l'origine, on pourrait utiliser la distance par rapport à un autre point. Voir Annexe C.

Note 5

La référence à l'entrepôt est implicite, ce qui est rendu possible par son unicité. Le caractère implicite n'est donc pas spécifique aux référentiels. Voir Annexe D.

2.2. Relativisation dense

La densification de la relation utilisera des ensemble non vides de points contigus appelés intervalles.

Exemple

Un sous-type d'intervalles IT de points de sous-type est déclaré comme suit :

```
INTERVAL IT OF PT
```

Ce qui est l'abréviation commode de

```
SUBTYPE IT OF TUPLE {begin PT, end PT}
  CONSTRAINT (begin ≤ end)
```

R@i, la relativisation dense de R, est alors définie comme suit :

```
RELVAR R@i {k KT, a AT, @i IT}
  KEY {k} RELATIVE TO (@i)
  PREDICATE Q(k,a) ∧ RDR[] ∈ @i
```

Où `KEY {k} RELATIVE TO (@i)` est équivalent à

```
CONSTRAINT R@i_No_contradiction :
  WITH S := UNFOLD R@i ON (@i) :
    CARD(S) = CARD(S PROJECT {k,@i})

CONSTRAINT R@i_No_redundancy :
  IS_EMPTY (R@i JOIN (R@i RENAME {@i AS @j}) WHERE END(@i) > PRE(@j) AND @i <> @j)

CONSTRAINT R@i_No_circumlocution :
  IS_EMPTY (R@i JOIN (R@i RENAME {@i AS @j}) WHERE END(@i) = PRE(@j) AND @i <> @j)
```

La valeur relativisée dense correspondant à la valeur non dense de la sous-section précédente est la suivante :

```
R@i{
  {k1, a1, [4,6]},
  {k2, a2, [4,4]},
  {k1, a2, [7,7]},
  {k3, a1, [4,5]},
  {k3, a3, [6,6]},
  {k3, a1, [7,7]}
}
```

Note 1

Les contraintes R_No_redundancy et R_No_circumlocution peuvent facilement être fusionnées en une seule :

```
CONSTRAINT R@i_No_redundancy_no_circumlocution :
  IS_EMPTY (R@i JOIN (R@i RENAME {@i AS @j}) WHERE END(@i) >= PRE(@j) AND @i <> @j)
```

Note 2

Les définitions de FOLD (PACK), UNFOLD (UNPACK) et NORMALIZE (USING) sont disponibles dans [Lorentzos 1994] ([Date 2014]).

Note 3

On remarque qu'il existe une bijection entre les points et les intervalles unitaires, donc entre R@p et sa transposition à l'aide d'intervalles unitaires R@u :

```
RELVAR R@u {k KT, a AT, @u IT}
  KEY {k,@u}
  CONSTRAINT R@u_Unicity : (@u) = 1
  PREDICATE Q(k,a) ∧ RDR[] ∈ @u
```

Les conversions sont triviales :

```
R@u := (R@p EXTEND {@u := [@p,@p]}) PROJECT {ALL BUT @p}
R@p := (R@u EXTEND {@p := END(@u)}) PROJECT {ALL BUT @u}
```

Pour cette raison, les fonctions FOLD et UNFOLD sont définies usuellement en regard de relations dont les attributs associés aux référentiels sont des intervalles et non des points.

Note 4

Il peut y avoir plusieurs degrés de densité, on ne s'intéresse ici qu'aux degrés extrêmes, le plus dense (FOLD) et le moins dense (UNFOLD). Le moins dense correspond à la «ponctualisation» du prédicat d'origine à une bijection près. La plus dense est obtenu par l'application de la non-circonlocution après remplacement des points par des intervalles (c'est le travail de FOLD). La non-redondance du résultat est corolaire de la définition même de la relation (puisqu'elle est un ensemble de tuples) et la non-contradiction est imposée par de la contrainte de clé, le cas échéant.

Note 5

Les contraintes R_No_redundancy et R_No_contradiction ne sont pas équivalentes. Voici un contre-exemple

```
Contradiction                Redondance

R@i{                         R@i{
  {k1, a1, [4,5]},             {k1, a1, [4,5]},
  {k1, a2, [5,6]}              {k1, a1, [5,6]}
}                            }
```

¿¿La note 5 est-elle vraiment utile ??

Note 6

Le fait qu'un intervalle ne puisse être vide est fréquemment perçu comme «agaçant» et donc débattu. Pour une discussion à cet égard, voir l'annexe E.

Note 7

Sur la base de la bijection entre point et intervalle singleton, faut-il permettre la conversion implicite entre eux ? La question est débattue et relève plus du langage que du modèle relationnel. Afin mieux présenter les choix de modélisation et les conséquences de ces choix, dans le présent document, sauf indication contraire explicite, il ne sera pas fait recours à la conversion implicite.

2.3. Relativisation dense avec horizon

2.3.1. Motivation

Nous proposons de généraliser les opérateurs FOLD, UNFOLD et NORMALIZE en permettant de restreindre la portée des référentiels à un horizon paramétrable au moment de calculer les expressions relationnelles, d'en densifier ou non la représentation, etc. Ceci permettra plus de flexibilité tout en évitant d'intégrer une sémantique spécifique à un type particulier de référentiels.

¡¡ À compléter !!

¡¡ Justification et exemples avec portions de route, épisodes de soin, etc. !!

2.3.2. Généralisation de FOLD, UNFOLD et NORMALIZE

Soit un attribut @i de type intervalle (ici IT) représentant un référentiel et un horizon, une valeur d'intervalle (ici @h) de ce même type, la relativisation dense d'une relvar R en regard de cet horizon est dénotée par

```
NORMALIZE R ON (@i:@h)
```

En l'absence d'horizon explicite, les bornes du type point définissant le type intervalle de l'attribut sont utilisées (dans notre exemple, [first,last] ; en général, pour un type intervalle T, [MIN(T),MAX(T)]).

Par extension toute expression dénotant une valeur de relation peut être normalisée, par exemple, pour un opérateur relationnel binaire interne <op> tels JOIN (⋈, jointure), UNION (∪, union), EXCEPT(−, différence) :

```
NORMALIZE R1 <op> R2 ON (@i:@h)
```

est équivalent à

```
FOLD (UNFOLD R1 ON (@i:@h)) <op> (UNFOLD R2 ON (@i:@h)) ON (@i:@h)
```

Il en va de façon analogue pour les autres opérateurs relationnels tels PROJECT (π, projection), WHERE (σ, restriction), EXTEND (ξ, extension).

Dans la portée d'une instruction FOLD, UNFOLD et NORMALIZE, l'horizon courant @h est obtenu par la fonction CURRENT_HOR.

Note 1

En général, dans UHF 1.0, les bornes de l'horizon étaient désignées par des fonctions prédéfinies à connotation temporelle. Nous leur préfèrerons désormais une dénotation plus neutre

- *saw* (since a while) est désormais défini par BEGIN(CURRENT_HOR).
- *ufn* (until further notice) par END(CURRENT_HOR).

Note 2

La relativisation dense avec horizon est aussi appelée «normalisation». L'horizon est aussi appelé portée, étendue, *portion* et *reférence*.

2.3.3. Transposition à la solution proposée par UFH 1.0

UHF 1.0 a étendu la solution présentée par DDL en ouvrant les intervalles non seulement vers la fin (until further notice - ufn - since) mais aussi vers le début (since a while - saw - until). En traitant le modèle UHF 1.0, nous traitons donc également du modèle proposé par DDL.

La relativisation dense avec horizon selon UHF 1.0 se présente donc comme suit (pour la définition des opérateurs NORMALIZE, FOLD et UNFOLD en regard de la variété des partitions, voir l'annexe A) :

```
RELVAR R@bx {k KT, a AT, @bx PT}
  KEY {k}
  PREDICATE (PREDICATE_OF[R](k,a) ∧ (@bx <= RDR[] <= END(CURRENT_HOR))
RELVAR R@be {k KT, a AT, @be IT}
  KEY {k, @be} RELATIVE TO (be)
  PREDICATE (PREDICATE_OF[R](k,a) ∧ RDR[] ∈ @be)
RELVAR R@xe {k KT, a AT, @xe PT}
  KEY {k}
  PREDICATE (PREDICATE_OF[R](k,a) ∧ (BEGIN(CURRENT_HOR) <= RDR[] <= @xe)

CONSTRAINT R_No_contradiction
  IS_EMPTY ((R@bx RENAME {a AS a_bx}) JOIN (R@be RENAME {a AS a_be})
    WHERE @bx < POST(@be) AND a_bx <> a_be)  AND
  IS_EMPTY ((R@be RENAME {a AS a_be}) JOIN (R@xe RENAME {a AS a_xe})
    WHERE @xe > PRE(@be)  AND a_be <> a_xe)  AND
  IS_EMPTY ((R@xe RENAME {a AS a_xe}) JOIN (R@bx RENAME {a AS a_bx})
    WHERE @xe > PRE(@bx)  AND a_xe <> a_bx);

CONSTRAINT R_No_redundancy
  IS_EMPTY (R@bx JOIN R@be WHERE @bx < POST(@be)) AND
  IS_EMPTY (R@be JOIN R@xe WHERE @xe > PRE(@be))  AND
  IS_EMPTY (R@xe JOIN R@bx WHERE @xe > PRE(@bx));

CONSTRAINT R_No_circumlocution
  IS_EMPTY (R@bx JOIN R@be WHERE @bx = POST(@be)) AND
  IS_EMPTY (R@be JOIN R@xe WHERE @xe = PRE(@be))  AND
  IS_EMPTY (R@xe JOIN R@bx WHERE @xe = PRE(@bx));

RELVAR R@v {k K, a AT @v IT} :=
  NORMALIZE
    R@bx RENAME {@bx AS @v}
      UNION R@be RENAME {@be AS @v}
      UNION R@xe RENAME {@xe AS @v}
  ON (@v)
```

Faut-il corriger R@v en utilisant CURRENT_HOR ?

```
RELVAR R@v {k K, a AT, @v IT} :=
  NORMALIZE
    (R@bx EXTEND {@v := [@bx,END(CURRENT_HOR)]} PROJECT ALL_BUT {@bx})
      UNION (R@be RENAME {@be AS @v})
      UNION (R@xe EXTEND {@v := [BEGIN(CURRENT_HOR),@xe]} PROJECT ALL_BUT {@bx})
  ON (@v)
```

CURRENT_HOR est-il, doit-il, être défini en dehors de la portée d'un horizon explicite (défini par un NORMALIZE) ?

Note sur les clés référentielles (étrangères)

¡¡ À développer !!

3. Illustration du problème

Dans le modèle UHF 1.0, lorsqu'une opération conduit à avoir un intervalle ouvert à la fois en début et en fin, elle est refusée, car non représentable au sein des seules partitions xe, be et bx. Cette situation constitue un frein à la modélisation d'évènements courants, par exemple l'état d'une entité observée de façon continue sur une période sans toutefois que ledit observateur ait pu constater le début de cet état ni sa fin. En nous appuyant sur un exemple simple, nous illustrerons quand et comment de telles situations peuvent se produire. Nous proposerons ensuite l'ajout d'une nouvelle partition et montrerons comment elle permet de résoudre le problème.

3.1. Situation initiale

Notre exemple présente trois entités identifiées par les clés k1, k2 et k3 :

- Toutes trois ont été observées au point 4 et leur état (représenté par l'attribut a), consigné. Aucune autre observation n'est disponible antérieurement au point 4.
- Toutes trois ont été observées au point 7 et leur état, consigné. Aucune autre observation n'est disponible postérieurement au temps 7.
- Aucune observation intermédiaire entre 4 et 7 (temps 5 et 6) n'est disponible.

Ceci peut être représenté par

```
R@xe{
}
R@be{
  {k1, a1, 4},
  {k2, a2, 4},
  {k3, a4, 4},
  {k1, a1, 7},
  {k2, a2, 7},
  {k3, a3, 7}
}
R@bx{
}
```

Par contre, si nous le représentons comme suit

```
R@xe{
  {k1, a1, 4},
  {k2, a2, 4},
  {k3, a4, 4}
}
R@be{
}
R@bx{
  {k1, a1, 7},
  {k2, a2, 7},
  {k3, a3, 7}
}
```

L'interprétation est différente pour chacune des trois clés k1, k2 et k3 :

- Toutes trois ont été observées au point 4 et leur état (représenté par l'attribut a), consigné. Aucune autre observation n'*étant* disponible antérieurement au point 4, *elle peut être étendue jusqu'au point BEGIN(CURRENT_HOR)*.
- Toutes trois ont été observées au point 7 et leur état, consigné. Aucune autre observation n'*étant* disponible postérieurement au temps 7, *elle peut être étendue jusqu'au point END(CURRENT_HOR)*.
- Aucune observation intermédiaire entre 4 et 7 (temps 5 et 6) n'est disponible.

3.2. Insertion des tuples

En regard de la précédente représentation, examinons l'impact de l'ajout de trois observations couvrant la période [5, 6] : {k1, a1, [5,6]}, {k2, a8, [5,6]} et {k3, a3, [5,6]}

```
R@xe{
  {k1, a1, 4},      -- #1a
  {k2, a2, 4},
  {k3, a4, 4}
}
R@be{
  {k2, a8, [5,6]}   -- #2
}
R@bx{
  {k1, a1, 5},      -- #1b
  {k2, a2, 7},
  {k3, a3, 5}       -- #3
}
```

Note 1

La tentative d'insertion de {k1, a1, [5,6]} créée une situation ambigüe : elle pourrait avoir lieu tout autant dans @be (#1a) que dans @xe (#1b). Dans un cas comme dans l'autre, l'insertion ne peut cependant être complétée en raison de la contrainte R_No_circumlocution. On remarque que l'état de k1 est le même dans @bx, @xe et dans la nouvelle observation. En fait, il faut refléter que l'état de k1 est demeuré constant (a=a1) :

- pour tous les points de l'intervalle [4,7] ;
- que, sans autre information, cet état demeure le même pour les points inférieurs à 4 ;
- que, sans autre information, cet état demeure le même pour les points supérieurs à 7.

Note 2

La tentative d'insertion de {k2, a8, [5,6]} n'est pas ambigüe et l'insertion se fait dans @be (#2). La contrainte R_No_circumlocution est respectée. Bien que l'état de k2 est le même dans @bx et @xe (a=a2), la nouvelle

observation brise la continuité par le nouvel état (a=a8), ce qu'on peut constater plus aisément en considérant la version ponctuelle :

```
R@p{
  {k2, a2, 4},
  {k2, a8, 5}, # rupture avec ce qui précède, @p=4
  {k2, a8, 6}, # rupture avec de qui suit, @p=7
  {k2, a2, 7}
}
```

Note 3

La tentative d'insertion de {k3, a3, [5,6]} n'est pas ambigüe l'insertion se fait dans @bx (#3). La contrainte R_No_circumlocution est respectée. L'état de k3 n'étant pas le même dans @bx (a=a3) et @xe (a=a4), bien que la nouvelle observation poursuive la continuité avec de l'une d'entre elles, il y a une rupture avec l'autre.

3.3. Pourquoi parle-t-on de perte de données ?

Considérons la valeur de R@v, avant insertion de {k1, a1, [5,6]} et pour les seuls tuples de clé k=k1 :

```
R@v{
  {k1, a1, [BEGIN(CURRENT_HOR),4]}, -- provenant de @xe
  {k1, a1, [7,END(CURRENT_HOR)]}    -- provenant de @be
}
```

Après insertion de {k1, a1, [5,6]}, hors densification, on obtient :

```
R@v{
  {k1, a1, [BEGIN(CURRENT_HOR),4]},
  {k1, a1, [5,6]},
  {k1, a1, [7,END(CURRENT_HOR)]}
}
```

Sous la forme dense (pour un BEGIN(CURRENT_HOR)<4 et un END(CURRENT_HOR)>7 déterminés), on obtient :

```
R@v{
  {k1, a1, [BEGIN(CURRENT_HOR),BEGIN(CURRENT_HOR)]},
  ...
  {k1, a1, [4,4]},
  {k1, a1, [5,5]},
  {k1, a1, [6,6]},
  {k1, a1, [7,7]},
  ...
  {k1, a1, [END(CURRENT_HOR),END(CURRENT_HOR)]}
}
```

Dans un tel contexte, la mise à jour de l'attribut a (de a1 à a2) au point 5 pour la clé k1 doit résulter en ceci :

```
R@v{
  {k1, a1, [BEGIN(CURRENT_HOR),BEGIN(CURRENT_HOR)]},
  ...
  {k1, a1, [4,4]},
  {k1, a2, [5,5]},
```

```
  {k1, a1, [6,6]},
  {k1, a1, [7,7]},
  ...
  {k1, a1, [END(CURRENT_HOR),END(CURRENT_HOR)]}
}
```

Ce qui se traduira, sous forme dense selon les partitions appropriées, en ceci :

```
R@xe{
  {k1, a1, 4}
}
R@be{
  {k1, a2, [5,5]}
}
R@bx{
  {k1, a1, 6}
}
```

En l'absence de la conservation de l'intervalle [4,7], il n'est plus possible de restaurer les bornes correctement, puisqu'on a perdu les bornes d'origine. C'est ce qui motivera l'introduction d'une nouvelle partition (@df) dans le prochain modèle (dit UHF 1.1).

Par ailleurs, on peut se demander s'il n'est pas plus « adéquat » de conserver tous les points pour lesquels une assertion explicite a été faite. Dans notre exemple, comme dans le modèle de Date, nous "perdons" la proposition relative au point 7. Le modèle UHF 1.2 explorera cette question.

4. Relativisation avec agents

¡¡ À rédiger !!

5. Modèle UHF 1.1

La définition des versions 1.1 et 1.2 de UHF ainsi que les annexes A, B, C, D et E sont en cours de rédaction.

Références

Date, C.J., Darwen, H., Lorentzos, N.A.: Temporal data and the relational model: a detailed investigation into the application of interval and relation theory to the problem of temporal database management. Morgan Kaufmann Publishers, San Diego, CA (2003).

Date, C.J., Darwen, H., Lorentzos, N.A.: Time and Relational Theory: Temporal Databases in the Relational Model and SQL. Morgan Kaufmann, Waltham, MA (2014).

Khnaisser, C., Lavoie, L., Burgun, A., Ethier, J.-F.: Past Indeterminacy in Data Warehouse Design. Database and Expert Systems Applications. pp. 90–100 Springer, Cham (2017).

Lorentzos, N.A.: The Interval-extended Relational Model and Its Applications to Valid-time Databases. Temporal Databases. pp. 67–91 (1993).

Lorentzos, N.A., Johnson, R.G.: An extension of the relational model to support generic intervals. In: Schmidt, J.W., Ceri, S., and Missikoff, M. (eds.) Advances in Database Technology—EDBT '88. pp. 528–542 Springer Berlin Heidelberg (1988).

Lorentzos, N.A., Johnson, R.G.: Extending relational algebra to manipulate temporal data. Information Systems. 13, 3, 289–96 (1988).

Lorentzos, N.A., Johnson, R.G.: Requirements Specification for a Temporal Extension to the Relationsl

Model. IEEE Data Eng. Bull. 11, 4, 26–33 (1988).

Lorentzos, N.A., Johnson, R.G.: TRA: A Model for a Temporal Relational Algebra. In: Rolland, C., Bodart, F., and Léonard, M. (eds.) Temporal Aspects in Information Systems, Proceedings of the IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems, Sophia-Antipolis, France, 13-15 May, 1987. pp. 95–108 North-Holland / Elsevier (1987).

Lorentzos, N.A., Mitsopoulos, Y.G.: SQL extension for interval data. IEEE Transactions on Knowledge and Data Engineering. 9, 3, 480–499 (1997).

Lorentzos, N.A., Poulovassilis, A., Small, C.: Implementation of Update Operations for Interval Relations. Comput. J. 37, 3, 164–176 (1994).

Lorentzos, N.A., Poulovassilis, A., Small, C.: Manipulation operations for an interval-extended relational model. Data & Knowledge Engineering. 17, 1, 1–29 (1995).

Tzouramanis, T., Manolopoulos, Y., Lorentzos, N.: Overlapping B+-trees: An implementation of a transaction time access method. Data & Knowledge Engineering. 29, 3, 381–404 (1999).

Vassilakis, C., Lorentzos, N.A., Georgiadis, P.: Implementation of Transaction and Concurrency Control Support in a Temporal DBMS. Inf. Syst. 23, 5, 335–350 (1998).

Vassilakis, C., Lorentzos, N.A., Georgiadis, P.: Transaction Support in a Temporal DBMS. In: Clifford, J. and Tuzhilin, A. (eds.) Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17-18 September 1995. pp. 255–271 Springer (1995).

Viqueira, J.R.R., Lorentzos, N.A.: SQL Extension for Spatio-temporal Data. The VLDB Journal. 16, 2, 179–200 (2007).

Produit le 2025-02-11 06:14:32 -0500

Groupe de recherche en modélisation et gestion de données