TEMPOS : un modèle d'historiques pour un SGBD temporel

Jean-François Canavaggio

Thèse

présentée par

Jean-François Canavaggio

pour obtenir le titre de

Docteur de l'Université Joseph Fourier - Grenoble 1

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

spécialité

Informatique

Tempos : un modèle d'historiques pour un SGBD temporel

Date de soutenance : 22 novembre 1997

Composition du jury :

Président du jury :	M. Michel Adiba
Rapporteurs :	Mme Colette Rolland
	M. Jacques Le Maitre
Examinateur :	M. Willy Goldgewicht
Directeurs de thèse :	M. Pierre-Claude Scholl
	Mlle Marie-Christine Fauvet

Thèse préparée au sein du laboratoire LogicielsΓSystèmes et RéseauxΓLSR-IMAG

Cette thèse n'est pas la mienne.

Du moins, je ne peux prétendre qu'elle corresponde seulement à mon travail. Marie-Christine et Pierre-Claude ont commencé à défricher le domaine avant que je ne commence ma thèse. Puis c'est ensemble que nous avons progressé dans la définition du modèle du temps, puis du modèle d'historiques. Leurs idées comme les miennes, nos discussions sont à la base de ce travail. Je ne peux que les remercier infiniment pour le temps et l'énergie qu'ils ont consacrés à notre travail d'équipe, et plus encore pour la confiance qu'ils m'ont toujours témoignée.

Marlon, arrivé plus récemment, s'est également beaucoup impliqué dans le projet, et je le remercie particulièrement du travail qu'il a réalisé pour l'implantation du prototype.

Marie-Claude nous a aidés à appliquer les historiques au cas des séries chronologiques. Sa bonne humeur a toujours largement compensé les questions embêtantes qu'elle posait, mais, de toute façon, celles-ci ont souvent permis d'éclaircir des points litigieux de notre modèle.

Mon séjour à Thomson/RCC a été très enrichissant, le travail et les discussions avec Willy Goldgewicht et Renaud Pommepuy ont été fort intéressants. L'environnement était très différent de celui du labo. Je les remercie tous deux pour leur accueil pendant ce stage, et je suis reconnaissant à Willy d'avoir accepté de participer à mon jury.

Je tiens à remercier M. Jacques Le Maitre, pour avoir accepté d'être rapporteur de ce travail et aussi pour m'avoir permis de l'exposer en détail dans son laboratoire.

Je remercie Mme Colette Rolland, qui a également accepté le rôle de rapporteur, et M. Michel Adiba, pour leur participation à ce jury.

Liliane Di-Giacomo, Martine Pernice, Solange Roche, François Challier et Bernard Martinet (bref, le "personnel administratif" du LSR) sont tout autant efficaces qu'aimables. Merci à eux pour les dossiers remplis et les pannes résolues!

En bon apprenti enseignant-chercheur, j'ai passé une partie de mon temps à enseigner. Aussi, je suis reconnaissant à tous les étudiants qui se sont vus imposer ma présence, qui ont eu à essuyer les plâtres de mes premières heures d'enseignement et qui n'ont pas porté plainte!

En créant le Collectif des Doctorants Grenoblois (CDG), nous avons voulu sensibiliser "les acteurs de la recherche", du monde académique et de l'industrie, aux problèmes rencontrés par les doctorants. La tâche est peut-être trop grande pour les quelques irréductibles du CDG, mais nous résistons encore. Nous n'avons pas trouvé la formule magique pour faire adhérer à notre action les thésards grenoblois, apparemment plus sensibles au ron-ron ambiant. Les personnes qui nous ont écoutés, et aidés, n'en sont que plus honorables.

Cependant, il n'y a pas que la thèse dans une vie de thésard (du moins, c'est ma thèse). Et si la recherche est une activité intéressante, elle est, à forte dose, aussi délétère que d'autres. Il a fallu plus d'un bouquin, et plus d'une journée de VTT pour me permettre de garder les neurones en place, et les pieds sur terre. La question n'est pas de savoir si Tempos est plus "performant" que TSQL 2, mais de savoir si le développement à tout va des outils informatiques est un gain pour les six milliards de terriens, ou seulement pour Bill Gates et ses semblables.

Je laisse à chacun le soin de réfléchir à cette question, et pendant ce temps, j'en profite pour remercier ceux qui ont partagé avec moi tous ces "à-côtés". J'ai une pensée particulière pour Hassen qui n'a pas bénéficié de conditions aussi favorables que les miennes et à qui on n'a pas laissé la possibilité de finir sa thèse.

Enfin, un dernier et grand merci à mes parents, pour leur aide pendant toutes ces années, mais aussi, et surtout, pour le regard sur le monde qu'ils m'ont donné.

Table des matières

Table des figures

Notations

Symbole	Signification
TYPE T	définition du type T
TYPE T1 (T2)	définition du type T1 paramétré par le type T2
<T1, T2, ..., Tn>	constructeur de n-uplets
{T}	constructeur d'ensembles
[T]	constructeur de séquences
T1 $\longrightarrow$ T2	constructeur de fonctions
λx1Γx2Γ.. $\bullet$ E	valeur de fonctions : λ introduit les noms des paramètres x1, x2, .. $\bullet$ introduit l'expression E décrivant la fonction
Nil	valeur absente
{ *commentaires* }	commentaire
Instant (u)	instant observé à l'unité u
Intervalle (u)	intervalle d'instants observés à l'unité u
D_séquence (u)	D_séquence d'instants observés à l'unité u

Introduction

Prise en compte du temps

Les systèmes de gestion de bases de données (SGBD) traditionnels offrent une vision instantanée du monde modélisé. L'état de la base est défini par la dernière mise à jour. L'évolution des données au fil du temps n'y est pas représentée dans la base de données. Lorsque la prise en compte de cette évolution est nécessaire, l'historique des données doit être géré au niveau des programmes d'application, ce qui en accroît la complexité. De plus, l'expression et la manipulation de valeurs temporelles est souvent très limitée. Le seul type date est souvent là pour satisfaire à tous les besoins, excluant d'office d'autres types de valeurs temporelles, comme les durées ou les intervalles. De même, le niveau de précision est couramment fixé au jour, ou à la seconde, mais il est impossible de prendre en compte des repères plus spécifiques, comme par exemple, les semestres, ou les demi-journées, etc.

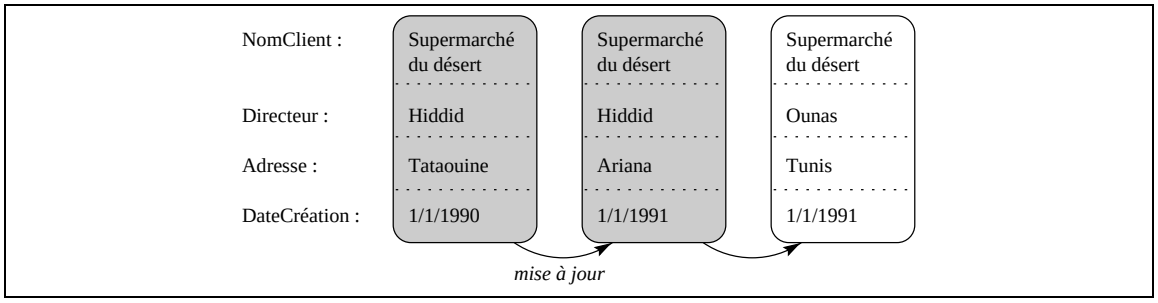


FIG. 0.1 – *Un exemple de base de données "classique"*

La figure 0.1 montre la valeur d'un objet dans une base de données "classique" ; il s'agit d'une fiche client d'une entreprise. La version courante de la base correspond au dernier bloc. Les blocs grisés correspondant à de précédents états de la base.

L'utilisateur peut interroger la base. Des requêtes possibles sont :

- Où est situé le client "Supermarché du désert" ?
- Qui a été le directeur du client "Supermarché du désert" après Mr Hiddid ?

Le SGBD ne peut répondre à cette deuxième requête : seul l'état courant est stocké dans la base de données.

Les bases de données temporelles visent à combler ces lacunesΓet permettent en particulier de dater les informations et de gérer leur historique.

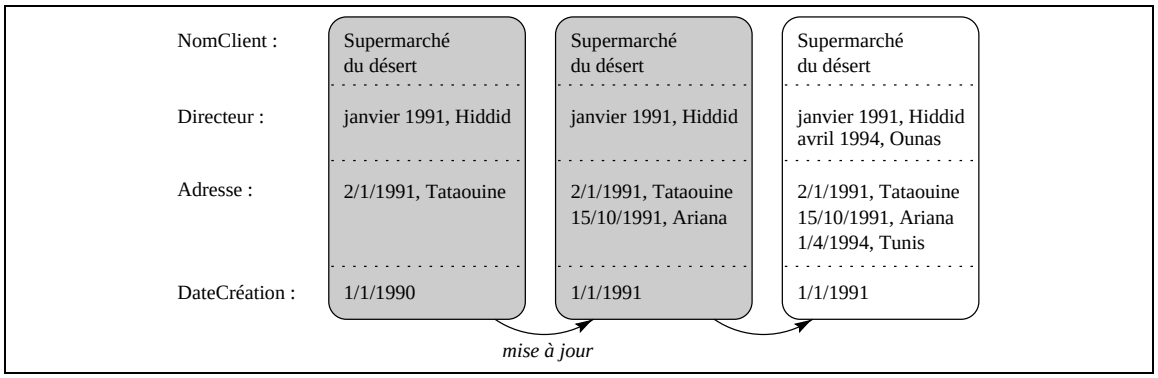


FIG. 0.2 – *Un exemple de base de données temporelle*

La figure 0.2 illustre l'exemple précédent dans le cadre d'une base de données temporelle. Dans ce casΓcertains attributs sont historisésΓalors que pour d'autresΓseule la dernière mise à jour est conservéeΓcomme dans une base de donnée "classique". Par exempleΓla première mise à jour indique que le "Supermarché du désert" est situé à Ariana depuis le 15/10/1991 (ce fait est daté et la valeur précédente conservée)Γet que la date de création de ce client est en fait le 1/1/1991 (la valeur précédente est perdue).

Il est maintenant possible de répondre à des requêtes telles que :

- Qui a été directeur après Mr Hiddid ?
- Combien de temps le client "Supermarché du désert" est-il resté à Tataouine ?

On peut remarquer dans cet exemple que les valeurs temporelles ne sont pas toutes exprimées à la même granularité : le jour pour l'adresseΓle mois pour le directeur. EnfinΓ cet exemple illustre différents types temporels : instants pour dater les informations mais également des duréesΓpour répondre à la deuxième requête.

Les bases de données temporelles

Les travaux relatifs aux bases de données temporelles visent à intégrer toutes les fonctionnalités nécessaires à la prise en compte du temps dans les SGBD.

Dans le domaine de la rechercheΓl'organisation de conférences spécialisées [TAI87Γ TIM95]Γla parution régulière de synthèses [Mac86ΓANC87ΓTAI87ΓSno88ΓSoo91ΓSno92Γ Sno93Γ Sno94aΓ FS95]Γun ouvrage [TCG+93] et de très nombreux articles montrent la variété des études qui ont été menéesΓleur prise en compte dans divers SGBD et la stabilisation progressive des concepts [JSS93ΓJCE+94].

En particulierΓdans le cadre des SGBD relationnelsΓla communauté scientifique est arrivée à la proposition de TSQL 2. La possible intégration de ces résultats à SQL 3 confirmera encore le modèle proposé comme standard (types temporels et langage de requêtes). Dans le cadre des SGBD à objetsΓles travaux sont plus hétérogènes.

Un point important des bases de données temporelles est l'accroissement de la complexité des requêtes. Cela se vérifie dès leur expression en langage naturel et cela augmente sensiblement avec la prise en compte de plusieurs dimensions temporelles. Il est donc nécessaire de définir des langages de haut niveauΓfacilement utilisables par les utilisateurs.

Contexte de travail

Le modèle TEMPOS présenté ici vise à étendre un SGBD à objets par les fonctionnalités nécessaires pour le rendre temporel. Il intègre en particulier les concepts et résultats déjà acquis dans les travaux actuels sur les bases de données temporellesΓqu'elles soient relationnelles ou à objets.

De plusΓil offre des niveaux d'abstraction adéquats à la mise en oeuvre de raisonnements temporels indépendants du type d'historique manipulé et de son implantation.

Il s'agit pour cela de définir un modèle temporel de donnéesΓde le formaliserΓde réaliser un prototype au dessus du SGBD O_2 [Tec95ΓDeu90] et de valider l'ensemble par diverses expérimentations.

Le modèle TEMPOS unifie les divers concepts nécessaires à la modélisation du temps et des historiques. Il est formalisé par la spécification fonctionnelle d'une hiérarchie de types. Sa validation est en coursΓd'un côté par la réalisation d'un prototypeΓet d'un autre côté par l'application du modèle proposé au cas des séries chronologiques.

Ce travail fait partie du projet STORM (Structure et Temporalité des Objets Réactifs Multimédias) [Gir95ΓAdi96ΓCC96ΓFCS97bΓFCS97aΓSFC98] qui traite plus largement de l'extension des SGBD à objets par de nouvelles fonctionnalités spécifiques aux aspects structurelsΓtemporels et actifs de données multimédias.

L'étude de la manipulation de données temporelles a fait l'objet d'un contrat avec la société Thomson/RCC. Une partie du prototype a été réalisé dans ce contexte. Nous avons par ailleursΓlors de cette collaborationΓconcrétisé notre étude dans le cadre d'une application industrielle confidentielle.

Organisation du document

Le chapitre 1 décrit un état de l'art des travaux du domaine des bases de données temporellesΓrelationnelles ou à objets en matière de représentation du temps et de modélisation des historiques.

Le chapitre 2 présente le modèle de représentation du temps de TEMPOS : notion d'unité d'observation du tempsΓtypes temporelsΓreprésentations multigranulaires de valeurs temporelles et manipulation de séquences d'instants. Le chapitre 3 est relatif au modèle d'historiques de TEMPOS : classification et caractérisation des historiquesΓtypes historiquesΓopérations de consultation et de mise à jour. Le chapitre 4 décrit l'implantation de TEMPOS réalisée au dessus du SGBD O_2.

L'annexe A décrit en détail une application (l'entreprise Oplate) où il est nécessaire de manipuler des historiques. Cet exemple nous sert à illustrer les notions présentées au cours de ce document. Cette annexe présente donc les différents éléments relatifs à l'application et propose quelques requêtes significatives dans ce contexte. Cet exemple est ensuite modélisé dans différents environnements (relationnels ou à objets) temporels ou non. Les requêtes sont exprimées dans les langages propres à chaque système.

L'annexe B présente une application de TEMPOS pour la modélisation de séries chronologiques.

L'annexe C définit les opérations de manipulation de séquencesΓque nous utilisons largement dans le modèle TEMPOS.

Chapitre 1

Modélisation temporelle d'une base de données

Une information temporelle est une association dans laquelle interviennent des *valeurs temporelles*Γdates ou périodesΓemployées pour marquer un événementΓpar exempleΓ"la société a ouvert le 1er janvier 1990"Γ"le contrôle technique doit avoir lieu tous les 2 ans". Les informations temporelles servent d'une part à mesurer des durées (la durée moyenne entre deux commandesΠla durée de validité d'une informationΠla durée de vie d'une entité dans une base de donnéesΓetc) et d'autre part à raisonner sur la succession entre événements ("les commandes du client le plus ancien") ou leur simultanéité ("quel était son adresse quand il a passé la commande numéro 49125?"). On peut ainsi déduire la validité de faits ("Oplate a 7 ans"Γ"le mois prochainΓil faut faire un contrôle")Γobserver l'évolution des informations dans le temps ("les ventes des 10 derniers mois")Γdéclencher des événements ("une facture doit être imprimée le dernier jour de chaque mois")Γcontrôler l'intégrité des données ("le prix d'une bouteille augmente si sa capacité augmente")Γetc.

Le temps apparaît comme une dimension spécifique dans la modélisation d'une base de données dès lors que l'on veut prendre en compte son évolution. Les valeurs temporelles intervenant dans une base de données jouent ainsi un rôle particulier pour dater les événements qui marquent la "vie" des composants observés : apparitionΓdisparition ou changement d'état.

Cette évolution peut concerner aussi bien les données que le schéma. Un schéma contient la définition des classesΓdes types d'une base de données. Une base contient des instances de ces classesΓreprésentant autant d'entités du monde réel. Les mécanismes d'évolution de schéma sont spécifiés [JCE+94] comme permettant de faire évoluer le schéma sans perte de données. Une synthèse des travaux dans ce domaine se trouve dans [Rod92ΓBen94ΓBF97].

Nous ne traitons ici que l'évolution des données.

L'*historique* (*history*[1]) d'un composant d'une base de données est ce que l'on veut retenir de son histoire : c'est-à-dire une suite chronologique des états considérés comme pertinents lors de son évolution dans le temps.

1.1 Informations temporelles

La question de la modélisation du temps est centrale dans les bases de données temporelles (voir par exemple : [CR87, MP93, WJS95]) et plus largement dans les systèmes d'information (voir par exemple [TAI87, CLR89, MBJK90, Cas93]). Sur un plan formel, les travaux s'appuient sur les diverses logiques temporelles (par exemple [Tur86]) qui fondent, entre autres, le raisonnement temporel en Intelligence Artificielle (voir par exemple [BL89, Ha91]) ou certaines méthodes de spécification et de vérification de programmes (voir par exemple [MP92]). De manière générale, le raisonnement temporel est basé sur la définition d'un domaine d'entités primitives (points ou intervalles, ou les deux), de relations d'ordre total ou partiel et dont les propriétés expriment les différences entre les modèles discrets, denses ou continus. L'état de l'art donné dans [Sch95] fournit une analyse des résultats utiles aux bases de données temporelles. De plus les modèles numériques incluent la notion de durée par la définition d'une métrique.

1.1.1 Valeurs temporelles

Les SGBD temporels visant à modéliser une évolution donnée d'un ensemble d'entités, la plupart des travaux s'appuient sur un modèle du temps linéaire et discret (Cf. figure 1.21). On notera toutefois que l'algèbre temporelle proposée dans [DBS96] est fondée sur un modèle continu et que dans TIGUKAT [GO93], la hiérarchie de types est organisée de manière à pouvoir traiter aussi des modèles arborescents discrets, denses ou continus [GLOS97].

Propositions	Densité	Linéaire/Arborescent
SQL 2	discret	linéaire
TSQL 2	discret	linéaire
[DBS96]	continu	linéaire
TFDL	discret	linéaire
ODMG	discret	linéaire
OSAM*/T	discret	linéaire
TF-ORM	discret	linéaire
TIGUKAT	discret, dense ou continu	linéaire ou arborescent
OODAPLEX	discret, dense ou continu	linéaire ou arborescent

FIG. 1.1 – *Caractéristiques du modèle du temps de diverses propositions*

1. Les termes anglais entre parenthèses sont ceux du glossaire [JCE+94], sauf indication spécifique.

Dans les systèmes d'information, on utilise en général un modèle discret. Cela conduit à définir le *degré de granularité* auquel on observe le temps. Chaque niveau de granularité fixe la taille des grains de temps que l'on manipule. Le *chronon* correspond aux grains les plus fins d'un système donné. Situer un événement à une granularité donnée, par exemple l'heure, signifie qu'à un niveau d'observation plus fin, par exemple la demi-heure, l'événement se produit à l'un des grains plus fins.

Le temps est ainsi vu comme un ensemble de grains de temps muni d'une relation d'ordre, que l'on peut schématiser par une droite sur laquelle chaque grain apparaît comme un point. De plus, on peut borner cette droite, par exemple en admettant l'existence de deux instants particuliers qui en marquent les limites. Enfin, comme l'on s'intéresse à l'évolution dans le temps, l'un des points représente l'instant présent, par rapport à la réalité ou par rapport au discours.

Une *durée* (*duration*) est une quantité de temps ("la production a été arrêtée 10 jours"). Cette notion est formalisée par la définition d'une fonction de distance.

Un *événement* (*event*) peut être situé dans le temps en *absolu* par son association à un grain de temps à un niveau de granularité donné, indépendamment de tout autre événement (*absolute time*), par exemple "le camion immatriculé 4856 VG 38 a été acheté le 12 février 1993". Mais un événement peut aussi être situé *relativement* à d'autres informations (*relative time*), par exemple "il a été acheté trois ans après le premier camion".

Une *période* (*span*) est un espace de temps plus ou moins long ("la période de validité d'un contrôle", "une période de sécheresse"). Cet espace est délimité par des événements particuliers (deux contrôles techniques, l'observation de seuils de débit de la source). Dans certains contextes une période caractérise un phénomène précis en général répétitif et sa durée est bien déterminée : "la période d'incubation", "la période de radioactivité d'un élément chimique", "la période d'un pendule", "la période d'une fonction", etc.

La droite du temps est communément découpée en diverses périodes de temps définies les unes par rapport aux autres : l'année est découpée en mois, le mois en jours, le jour en heures, etc, mais l'année est aussi découpée en semaines, et les semaines en jours. Certaines périodes ont des durées fixes (*fixed span*) : une heure vaut 60 minutes, une semaine vaut 7 jours, etc. D'autres périodes ont des durées variables selon le contexte (*variable span*) : une année vaut 366 ou 365 jours selon qu'elle est bissextile ou non, un mois vaut de 28 à 31 jours. Une période permet d'exprimer des temps relatifs ou de définir la périodicité d'un événement, par exemple "la paye est mensuelle", "l'entreprise est fermée en Août", etc.

Pour dénoter les valeurs temporelles, on se refère à un *calendrier* qui peut être lié à la culture, par exemple un calendrier grégorien ou un calendrier musulman, à un domaine d'application, par exemple un calendrier universitaire ou un calendrier fiscal, etc. Plus généralement, un *système calendaire* correspond à une évolution des calendriers dans le temps. Par exemple, en Occident, on est passé du calendrier romain au calendrier julien puis au calendrier grégorien, avec en France une courte période utilisant le calendrier révolutionnaire [Cou86].

Un calendrier définit les périodes de temps sur lesquelles il s'appuie d'une part pour offrir plusieurs niveaux de granularité et d'autre part pour fixer les conventions de représentation. Ces périodes permettent de nommer les grains aussi bien que de fixer le système d'unités dans lequel sont formulées les durées. En effetΓun grain est généralement caractérisé par le temps écoulé depuis un grain de référence. Un calendrier comporte aussi des règles d'usage comme celles qui fixent un changement d'heure selon la période de l'annéeΓ ou selon la situation géographique. Enfin un calendrier comporte une variété de formes d'expression des valeurs temporelles en tenant compte par exemple de la langue (1/2/97 change de sens en anglais ou en français) et établit la correspondance vis-à-vis d'autres calendriers (le jeudi de la semaine 52 en 1997 est aussi le 25/12/1997).

1.1.2 Multigranularité

Selon les objectifs assignés au SGDBΓcelui-ci doit permettre à l'utilisateur de s'exprimer selon les conventions définies par le calendrier de son choixΓau niveau de granularité qu'il désire. Au minimumΓle système doit s'appuyer sur une sémantique préciseΓsur des barrières d'abstraction adéquates distinguant les formes d'expression de valeurs temporelles et leur représentation et sur une architecture permettant des évolutions futures. Dans la forme la plus sophistiquée on doit permettre l'intégration de tout système calendaire et la cohabitation de plusieurs calendriers [SS92ΓCSS94].

Le degré de granularité peut être choisi différemment selon l'applicationΓou au sein d'une même application selon les événements considérés ou même encoreΓpour un événement donnéΓla granularité peut varier suivant le point de vue auquel on se place. La précision peut également être limitée par celle des instruments de mesure utilisés pour l'observation. Dès lors que des informations temporelles de granularités différentes sont confrontéesΓon doit préciser la sémantique des opérateurs mis en jeu (voir par exemple [Sar93]Γ§ 5.3.5). Pour préciser la correspondance entre deux niveaux de granularitéΓon doit pouvoir distinguer les cas où un événement ne se produit qu'à certains des grains plus fins (par exemple : "le contrôle technique doit être fait en Août") et les cas où l'événement se produit à tous ces chronons (par exemple : "le guichet est ouvert de 8h à 12h") [Kou94].

Nous illustrons quelques uns de ces besoins dans le cas de notre entreprise Oplate.

– *Un phénomène peut être considéré selon plusieurs points de vue* [Hob85ΓEuz94] :
  Par exempleΓles dates des contrôles techniques des camions sont stockées dans la base de données au niveau du jourΓCependantΓpour vérifier l'existence d'un contrôle (par exemple)Γl'année est un niveau d'observation plus adéquat : "y a-t-il eu un contrôle en 1995 ?". Le niveau maximal de détail n'est pas forcément utileΓet suivant le contexteΓ on ne souhaite qu'une approximation de la réalité : "il y a eu un contrôle en 1995" suffit comme réponseΓinutile de dire qu'il a eu lieu le 25 octobre.

– *Précision adaptée à l'information* :
L'évolution de la production des bouteilles est suivie à la granularité du jour. L'historisation de la production et celle des ventes permettent de calculer le stock journalier. Si ces observations étaient faites toutes les heures le stock calculé serait trop fluctuant (par exemple en fonction des heures de livraisons) et donc inexploitable. Inversement observés tous les mois l'usine ne serait pas capable de réagir à temps à une augmentation brutale des ventes par exemple.
Le débit de l'eau est observé toutes les heures. En effet il peut varier rapidement et il est nécessaire d'ajuster le rythme de production en conséquence.

Pour permettre la prise en compte de cette multigranularité plusieurs travaux proposent la notion d'unité de temps qui permet de donner un cadre formel à la manipulation de ces valeurs temporelles considérées à des granularités diverses.

[CR87] propose de décrire les unités temporelles par regroupement consécutifs de grains en partant des chronons. Les regroupements ne sont pas quelconques il s'agit de partitions successives (*constructed intervallic partition*). Cette approche ne fournit que des structures linéaires (*temporal universe*) et le cas des unités mois et semaines qui sont deux partitions différentes et non comparables des jours n'est pas pris en compte.

Dans [WJ93 WJS95] une unité temporelle (*time unit*) est une partition de l'ensemble des chronons (Cf. figure 1.2) définie par une fonction u de N dans 2^N ayant les propriétés suivantes :

- $\forall\ \mathsf{i} \in \mathrm{N},\ \exists\ \mathsf{j} \in \mathrm{N}$ tel que $\mathsf{i} \in \mathsf{u(j)}$
  { *Couverture du domaine temporel* }

- $\forall\ \mathsf{i1}, \mathsf{i2}, \mathsf{i3} \in \mathrm{N}$ tels que $\mathsf{i1}<\mathsf{i2}<\mathsf{i3}$, $\{\mathsf{i1},\mathsf{i3}\} \subset \mathsf{u(i)} \Rightarrow \mathsf{i2} \in \mathsf{u(i)}$
- $\forall\ \mathsf{i}, \mathsf{j} \in \mathrm{N},\ \mathsf{i} \neq \mathsf{j} \Rightarrow \mathsf{u(i)} \cap \mathsf{u(j)} = \emptyset$
  { *Partition du domaine en intervalles disjoints* }

- $\mathsf{0} \in \mathsf{u(0)}$
  { *Alignement des origines* }

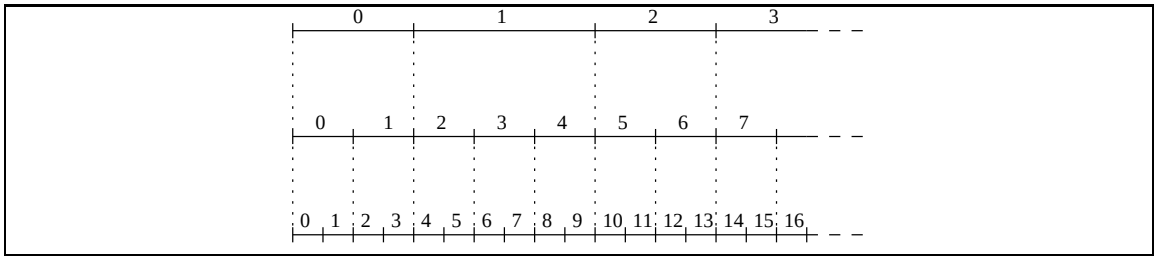


FIG. 1.2 – *Unité de temps : partition*

La définition précédente implique que toutes les unités commencent à l'origine, qu'un grain d'une unité est un morceau continu du temps, que deux grains ne se chevauchent pas et qu'y a pas de trou entre deux grains.

Une unité est *régulière* si tous ses grains se décomposent en un même nombre de chronons.

La figure 1.2 montrant bien la partition de la droite du temps peut à ce titre induire en erreur car la partition est visualisée par des intervalles d'une certaine longueur. La figure 1.3 montre la discrétisation du temps réalisée par le partitionnement des unités.

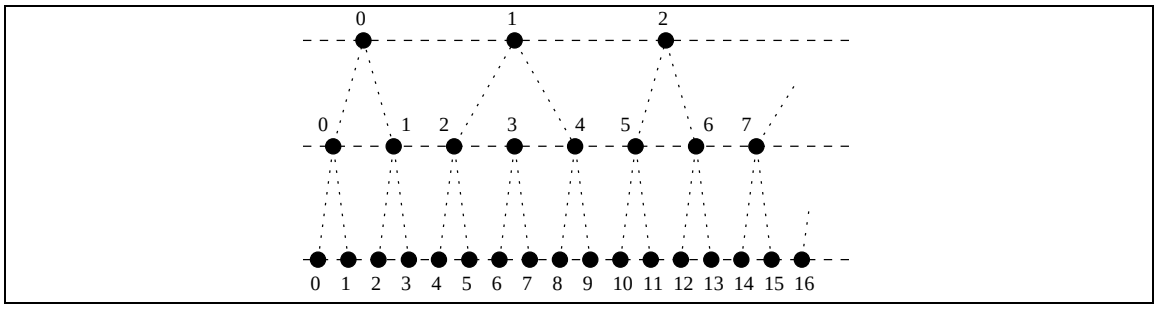


FIG. 1.3 – *Unité de temps : discrétisation*

Dans [CCMP93, BWJ96], on trouve une formalisation similaire où la droite du temps est décrite par domaines temporels (unités) ordonnés suivant leur granularité.

De plus, dans [CCMP93], les auteurs proposent, pour les opérations de conversions, de prendre des mesures de granularité constantes. Cela revient par exemple, à considérer des mois légaux de 30 jours.

[CD95] propose une démarche inverse où les unités sont décomposées en unités plus fines. Une propriété d'alignement garantit que les grains d'une unité coïncident avec les grains de l'unité supérieure. A partir d'un nombre réduit d'unités primitives, il est possible de définir de nouvelles unités à l'aide de composition ou décomposition.

1.1.3 Types temporels

Dans tous les travaux, durées, instants et intervalles sont des valeurs temporelles de base que l'on peut manipuler plus ou moins directement par des opérations standards (relations d'ordre, opérations arithmétiques, etc). Celles-ci sont largement détaillées dans le projet TEMPIS [SS92, SSD+92], également dans TSQL 2 [Sno95b]) et la figure 1.4 en présente un récapitulatif (il s'agit d'opérateurs binaires dont la première opérande est dans la colonne verticale et la seconde dans la colonne horizontale).

– Un *instant* (*instant, absolute time* dans [LBG93]) est un point particulier de la droite du temps. Il est considéré comme atomique et n'a pas de durée, relativement à son unité d'observation.

Exemples : le “14 juillet 1997” dans l'unité jourΓ“novembre 1997” dans l'unité mois.

On dispose de diverses opérations : égalitéΓrelation d'ordreΓdurée entre deux instantsΓ ajout ou retrait d'une durée à un instantΓetc. Des fonctions particulières dénotent les instants associés à l'horloge du système (*now*) et aux extrémités de la droite du temps (*beginning, forever*).

– Une *durée* (*spanΓ duration* dans [LBG93]) est un espace de temps. Une durée est parfois définie comme un intervalle non-ancré sur la droite du temps (donc caractérisé uniquement pas sa longueur)[2].

  Exemples : 4 heuresΓ2 mois.

  Il est possible de convertir des durées exprimées dans des unités régulières vers d'autres unités régulières. Par exempleΓune durée de 10 jours est égale à une durée de 240 heures. Par contreΓles durées exprimées dans des unités non régulières posent problème. En effetΓleur interprétation dépend du contexte : une année correspondΓselon le casΓà 365 ou 366 jours.

– Un *intervalle* (*time interval*) est un segment de la droite du tempsΓdélimité par deux instants précis ([Lad86] parle d'*intervalle convexe*). C'est un ensemble de grains adjacents deux à deux.

  Exemple : [1 juin 1997Γ15 juin 1997].

2. Ceci explique probablement le choix du nom (*interval*) du type SQL 2, ce qui génère une certaine confusion.

Op1 \ Op2	Instant	Durée	Intervalle	Elément temporel
Instant	Durée séparation Relations (=Γ<)	Décaler	Relations (avantΓaprèsΓ appartient)	Relations (avantΓaprèsΓ appartient)
Durée		Somme Différence		
Intervalle	Relations (avantΓaprèsΓ contient)	Décaler Allonger Raccourcir	Intersection Relations [All83]	Relations (avantΓaprèsΓ appartient)
Elément temporel	Ajout Retrait Relations (avantΓaprèsΓ contient)	Décaler Allonger Raccourcir (*un/tous les intervalles*)	Opérations ensemblistes Relations	Opérations ensemblistes Relations [Lad86]

FIG. 1.4 – *Opérations standards entre les divers types temporels*

La durée d'un intervalle est le nombre de grains qu'il comporte. On dispose par ailleurs des opérations ensemblistes usuelles et on peut les particulariser lorsque le résultat est un intervalle (voir par exemple [Sar93]Γ§ 5.3).

Il existe une variété de relations entre intervalles comme par exemple l'antériorité (un intervalle précède un autre)Γla rencontre (deux intervalles se joignent)Γle chevauchementΓl'inclusionΓla simultanéitéΓetc. (Cf. figure 1.5). Leur étude exhaustive est à la base de la logique temporelle proposée par Allen [All83] (voir aussi l'axiomatisation donnée dans [KJ93] fondée sur un ensemble discret d'intervalles et de points).

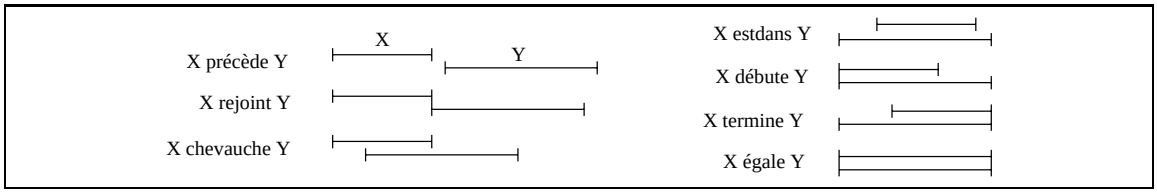


FIG. 1.5 – *Relations de Allen [All83]*

Des opérations particulières établissent le lien entre instants et intervallesΓles unes pour déterminer l'ensemble des instants d'un intervalle et les autres pour construire des intervalles par regroupement d'instants ou d'intervalles. Elles sont à la base d'opérations fondées sur la partition d'un ensemble d'informations temporelles (par exemple les n-uplets d'une relation) selon les valeurs des attributs non temporels et permettant des regroupements ou des expansions [BSS96] (*Fold-Unfold* [Lor93]Γ *Coalesce-Expand* [Sar93]]Γ*Nest-Unnest* [Tan93]).

– Un *élément temporel* (*temporal element* dans [JCE+94]Γ *coalesced temporal element* dans [BCTP95]Γ *interval* ou *non convex interval* dans [Lad86]Γ *generalized interval* dans [Yu 83]) est une union finie d'intervalles de temps disjoints. Cette notion permet de décrire des valeurs temporelles discontinuesΓpar exemple "les guerres mondiales"Γ ou dans le cadre d'OPlateΓ"les périodes où le débit de la source est inférieur à 1 m^3/s".

  Exemple : { [1914Γ1918]Γ[1939Γ1945] }.

  Les résultats de toutes les opérations de manipulation entre éléments temporels d'une partΓet instantsΓintervalles ou autres éléments temporels d'autre partΓsont des éléments temporels. En particulierΓles opérations ensemblistes permettent de combiner des éléments temporels entre eux. Un élément temporel peut être étendu (respectivement restreint) par ajout (respectivement par soustraction) d'instants ou d'intervalles. [Lad86] présente un ensemble de relations associées aux intervalles généralisésΓà partir de combinaisons des relations de Allen. La figure 1.6 présente quelques unes de ces relations.

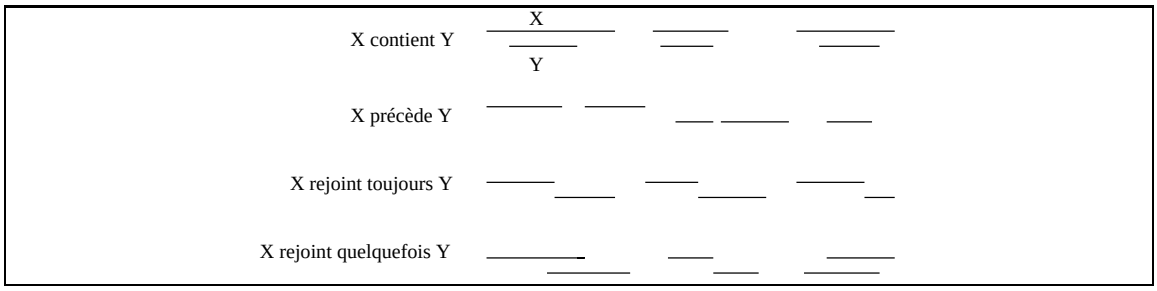


FIG. 1.6 – *Relations sur les intervalles non-convexes [Lad86]*

Le tableau de la figure 1.7 indique pour chacune des principales propositions les types temporels de base disponibles. Pour certains travaux il appelle quelques commentaires.

Propositions	Instants	Intervalles	Durées	Eléments temporels
Oracle	date		opérations avec les entiers	
SQL 2	date time timestamp	(date date)	interval	
TSQL 2	date time timestamp	period	interval	*temporal element*
O_2	date		opérations avec les entiers	
TF-ORM	instant	interval	span	
TIGUKAT	instant	interval	span	
OODAPLEX	point	region		{ point }
MCO	date	tranche de temps	délai	
Glossaire [JCE+94]	instant	time interval	span	temporal element

FIG. 1.7 – *Types temporels de base dans diverses propositions*

– **Oracle**

Le type `date` correspond à un instant exprimé dans les unités siècle/année/mois/jour/-heure/minute/seconde. Lorsque l'utilisateur fournit une date à une précision plus grossière que la seconde (par exemple juin 1997) le système affecte une valeur par défaut pour chaque unité omise (pour cet exemple : 1er juin 1997 12 h 0 min 0 s).

La notion de durée n'existe pas en tant que telle. Par contre il existe des opérations particulières qui permettent de considérer les entiers comme des durées dans une certaine unité. Par exemple `ADD_MONTHS(d, n)` ajoute `n` mois à la date `d`. Si le numéro de jour

n'est pas compatible avec la date obtenue (par exemple ADD_MONTHS('31/01/1997' ,1))Γ alors le résultat est le dernier jour du mois (iciΓle résultat est le '28/02/1997'). Il est possible de connaître la durée séparant deux datesΓOracle donne un entierΓqui correspond au nombre de jours de séparation.

- **O_2**

  Le type date correspond à un instant exprimé dans les unités année/mois/jour.

  Comme avec OracleΓla notion de durée n'existe pasΓmais des opérations permettent d'ajouter des joursΓdes mois ou des années en manipulant simplement des entiers.

  Il faut noter que le type date n'est pas un type primitif de O_2Γmais une classe de la boîte à outils standard. L'utilisateur peut donc définir une ou plusieurs autres classes plus sophistiquées.

- **SQL 2**

  Il existe plusieurs types correspondant aux instants. Ceux-ci se différencient par leur granularité. Par exempleΓle type date est à la granularité du jour alors que le type time est à la granularité de la seconde.

  Le type intervalle n'existe pas directement dans SQL 2. Par contreΓdes opérateurs (comme OVERLAPS) prennent en paramètre quatre instants et les manipulent comme deux intervalles (mais ceux-ci ne sont jamais construits en tant que tels).

- **TSQL 2**

  Le type timestamp permet d'exprimer un instant à n'importe quelle granularité. Les types date et time ne sont que des raccourcis syntaxiques et ne sont là que pour maintenir la compatibilité avec SQL 2.

  Le type temporal element n'existe pas en tant que tel parmi les types temporels définis en TSQL 2. Pourtant la période de validité (Cf. section 1.2) associée aux n-uplets dans les relations historiques est un élément temporel et le langage d'interrogation dispose d'opérations de manipulation de ces valeurs.

1.1.4 Calendriers

Un calendrier (*calendar*) est une interprétation particulière du temps [JCE+94]. C'est un ensemble de valeurs temporelles (instants) significatives pour un utilisateur. Le calendrier grégorienΓpar exempleΓfixe la décomposition du temps en annéesΓmoisΓjoursΓetc. On peut définir des calendriers d'utilisation plus restreinte pour parler par exemple des jours ouvrésΓdes premiers mardis de chaque moisΓetc. Ils s'agit de collections structurées d'instantsΓles éléments temporels en sont un exempleΓoffrant une structuration à un seul niveau.

La manipulation des calendriers est développée dans plusieurs travaux [LMF86, CSS94, CD95, LEW96] ; les opérations proposées se partagent en deux catégories. D'une part les opérations décrivant le découpage du temps, et celles-ci sont souvent proches de la description d'unités temporelles, évoquée dans la section 1.1.2.

D'autre part les opérations permettant de manipuler ces valeurs afin de les structurer (par exemple, découper un mois en jour) ou de les affiner (par exemple, ne garder que les mardis parmi les jours de la semaine, ou bien encore ne garder que le mois de janvier, etc). En combinant ces opérations entre elles, il est possible d'obtenir, par exemple, l'ensemble des mardis des mois de janvier. Nous les détaillons dans cette section.

Selon les travaux, les calendriers sont représentés par des éléments temporels ou bien par des collections d'instants ou d'intervalles à n niveaux (l'*ordre* d'une collection correspond à sa profondeur). Une collection d'ordre 2 est, par exemple, l'ensemble des mois d'une année, où, pour chaque mois on dispose de l'ensemble de ses jours (Cf. figure 1.8, cas d'une année non bissextile).

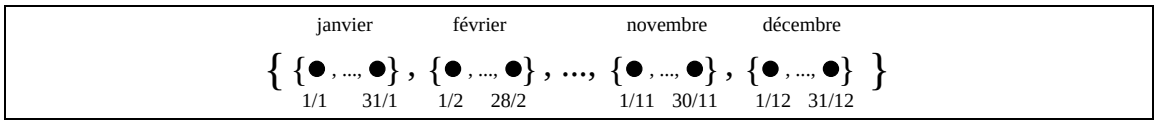


FIG. 1.8 – *Collection d'instants d'ordre 2*

Un opérateur de *décomposition* (*decomposition* dans [CD95], *dicing* dans [LMF86]) découpe un intervalle en des sous-intervalles, par exemple, une année en les mois qui la composent (Cf. figure 1.9).

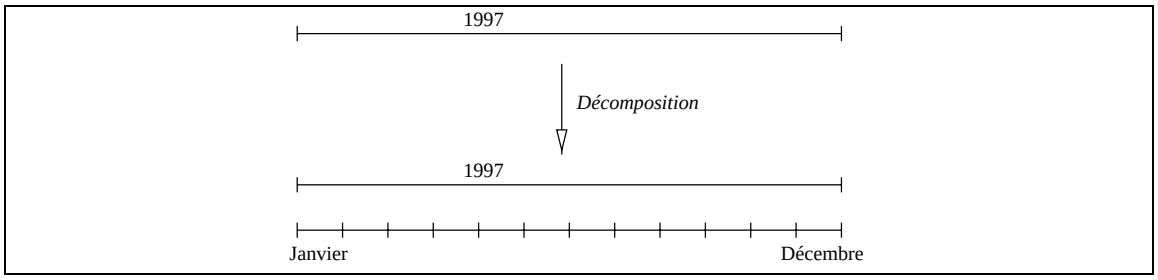


FIG. 1.9 – *Opération de décomposition*

Un opérateur de *sélection* (*slicing* dans [LMF86], *select* dans [LEW96]) permet de ne retenir que certains intervalles d'une collection, par exemple, les mardis de chaque semaine (Cf. figure 1.10).

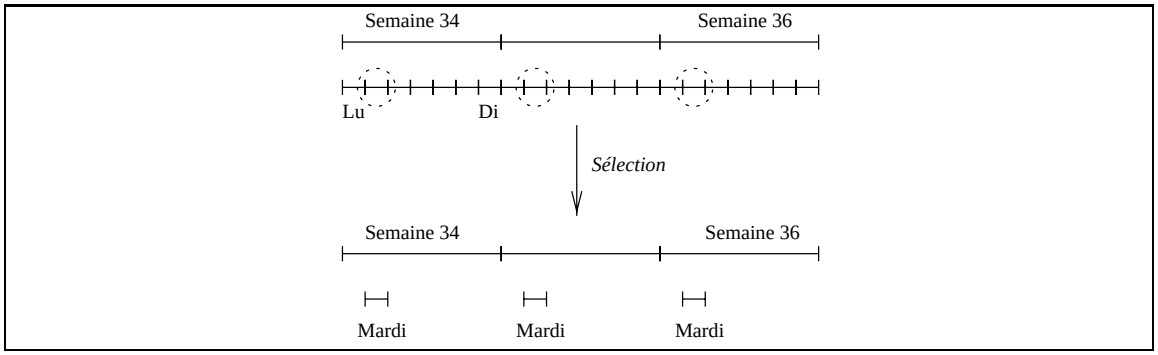


FIG. 1.10 – *Opération de sélection*

L'expression d'ensembles périodiques se fait ainsi par composition de ces opérations. Par exemple, pour obtenir les premiers jours de chaque mois de l'année 2000, il faut appliquer une décomposition de l'année en mois, puis une sélection sur les premiers jours du mois.

D'autres opérateurs plus classiques (comme l'union, l'intersection, l'exclusion) permettent également de manipuler des expressions sur les calendriers.

1.2 Historiques

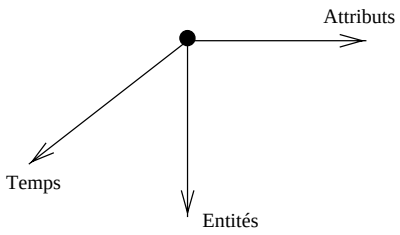
1.2.1 Caractéristiques structurelles et temporelles des entités

Pour modéliser l'évolution des entités du monde réel, il est utile de les étudier selon deux points de vue : leurs caractéristiques *temporelles* les situent dans le temps ; leurs caractéristiques *structurelles* fixent, au travers d'un ensemble d'attributs, le rapport entre les propriétés des entités telles qu'elles sont observées à un instant donné et les valeurs représentant ces propriétés dans la base.

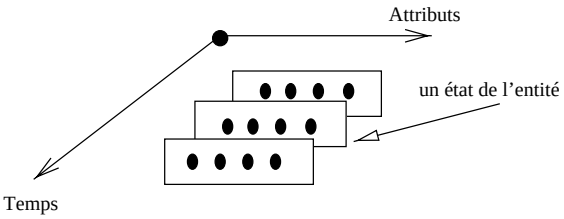
Remarquons qu'une valeur structurelle peut être d'un type temporel. C'est par exemple le cas de la date de mariage d'une personne, qui peut être enregistrée à un instant différent et qui peut, le cas échéant, être corrigée du fait d'une erreur lors de la saisie initiale.

La figure 1.11 schématise les caractéristiques structurelles et temporelles d'un ensemble d'entités de même type (schéma) selon trois axes : les entités considérées caractérisées par leur identité, leurs propriétés communes caractérisées par un ensemble d'attributs, et les instants considérés sur la droite du temps (Cf. figure 1.11.a). On peut alors considérer trois projections sur un plan, selon que l'on s'intéresse à l'*historique* d'une entité dans le temps (Cf. figure 1.11.b), ou à la *vision instantanée* (*snapshot*) d'un ensemble d'entités à un instant donné (Cf. figure 1.11.c), ou enfin aux *périodes d'observation* d'un ensemble d'entités (Cf. figure 1.11.d).

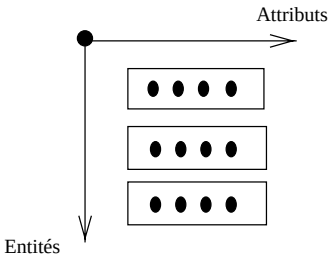
Selon l'application considérée, une propriété d'une entité peut être constante ou modifiable. Si elle est modifiable, on peut ne retenir que la dernière mise à jour ; ou bien



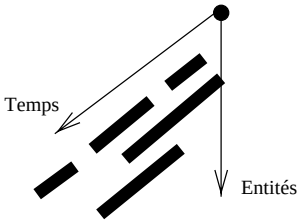
(a) Les différents axes d'observation



(b) Historique : une entité évolue dans le temps



(c) Instantané : un ensemble d'entités à un instant donné



(d) Périodes d'observation d'un ensemble d'entités

FIG. 1.11 – *Caractéristiques structurelle et temporelle des entités*

certains états pertinents de son évolution : l'attribut de l'objet correspondant est alors *historique*.

Lorsque le modèle comme par exemple TempSQL [GN93] impose que tous les attributs d'un objet ou d'un n-uplet d'une relation aient la même période de vie on parle d'*homogénéité temporelle* (*temporal homogeneity*). Cette propriété garantit l'absence de valeurs "nulles" dans les instantanés.

D'une entité à l'autre d'un attribut à l'autre les changements peuvent survenir à des instants observés à des niveaux de granularité différents. Dans notre exemple Oplate le débit de la source est observé toutes les douze heures alors que la production des bouteilles d'eau est observée journalièrement. Certains attributs peuvent évoluer de manière synchrone : leurs modifications surviennent aux mêmes instants par exemple si l'on impose que le prix d'une bouteille soit changé lorsque son volume change.

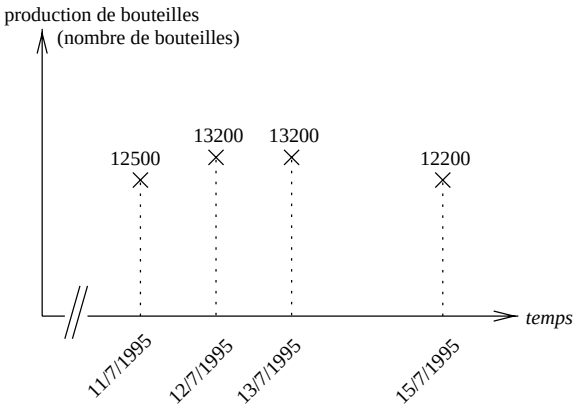
Un historique peut être vu comme une fonction d'un domaine temporel dans un domaine d'états ou comme une suite de couples <valeur temporelle état> ordonnée chronologiquement. [SS93] parle de *time sequence (TS)*. La signification des valeurs temporelles doit être précisée par le niveau de granularité d'observation du temps.

Un historique peut être représenté de diverses manières selon les contraintes éventuelles imposées par le modèle de données utilisé ou selon les propriétés particulières de l'historique considéré. On peut notamment distinguer les situations où il faut associer un état à toute valeur temporelle dans l'intervalle de validité (au niveau de granularité choisi) et les situations où l'on n'est intéressé que par certains points dans cet intervalle.

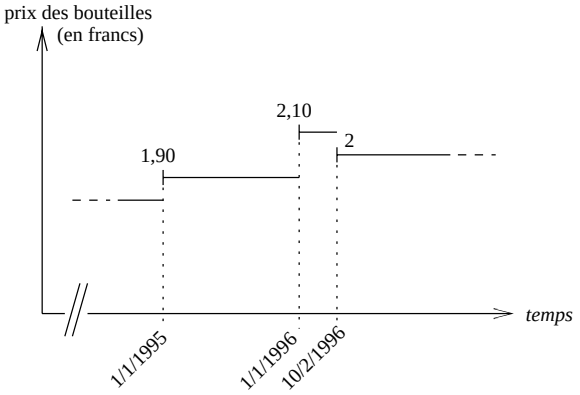
Dans notre exemple on peut considérer l'historique de la production de bouteilles l'historique des prix des bouteilles et l'historique du débit de la source. Dans le premier cas il n'y a pas de valeurs entre deux journées de production (week-end jour feriégrêve) la fonction historique n'est donc pas définie en dehors des points d'observation (historique discret *discrete TS* [SS93] Cf. figure 1.12a). Dans le deuxième cas l'historique est une fonction en escalier (*step wise constant TS* Cf. figure 1.12b) : la valeur est constante entre deux points d'observation ce qui est une forme simple d'interpolation. Enfin dans le troisième cas un modèle physique d'une source peut fournir la fonction d'interpolation adéquate entre deux mesures effectives (*continuous TS* Cf. figure 1.12c).

1.2.2 Dimensions temporelles

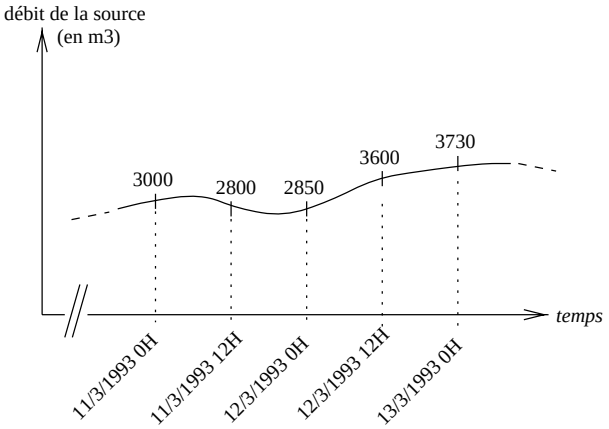
L'évolution d'une base de données peut être observée selon diverses interprétations des valeurs temporelles. A cet égard la distinction la plus notable concerne la différence entre le temps situant faits et événements dans le monde réel et celui situant les transactions opérant sur une base de données.



(a) Historique discret



(b) Historique en escalier



(c) Historique interpolé

FIG. 1.12 – *Différents types d'historiques*

Dans le glossaire du domaine [JCE+94]Γil est défini le temps de validité et le temps de transaction :

- Le temps de validité (*valid time*Γ*logical time* dans [DLW84]Γ*effective time* dans [Gad93]) (Cf. figure 1.13a) d'une donnée est le temps où la donnée est vraie (valide) dans le monde réelΓ par exemple "le camion a subi une vidange le 8/4/1997". Le temps de validité est fourni par l'application.

  Cette dimension permet de refléter le passé ou d'anticiper sur le futur dans la réalité du monde de l'application en se référant aux faits et événements à tout instant considéré comme significatif.

- Le temps de transaction (*transaction time*Γ *physical time* dans [DLW84]Γ *registration time* dans [Gad93]) (Cf. figure 1.13b) correspond au temps où une donnée est enregistrée dans la base de données.

  Le temps de transaction est fourni par le SGBD. Il peut s'agir du moment où la transaction associée est validée.

  En particulierΓun fait est rarement enregistré au moment même où il advient. La notion de délai structurel (*delay structure* dans [Ari87]) est une caractéristique des systèmes d'informations organisés où les informations ne sont enregistrées dans la base de données qu'un certain temps après avoir été effectivement connues.

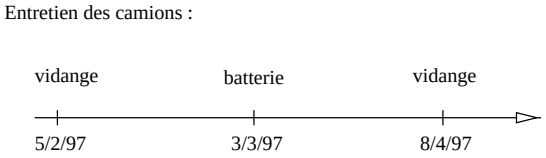
  Cette dimension traduit l'historique des modifications de la base de données. Elle permet d'observer l'état de la base à tout instant depuis sa mise en oeuvre que ce soit pour analyser les transactions d'une application ou pour assurer l'intégrité de la base ou pour assurer la fonction de reprise.

Une information de la base de données peut comporter ces deux types de valeursΓ qui éventuellement évoluent différemment dans le temps. On parle alors de valeur ou d'historique bitemporel (Cf. figure 1.13c).

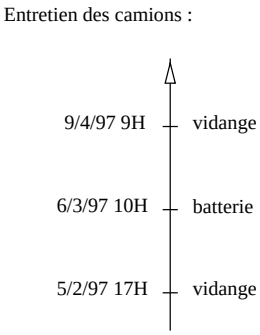
Ces deux dimensions temporelles ne sont pas similaires. Dans le cas du temps de validitéΓ les valeurs structurelles et temporelles peuvent être entâchées d'erreur. Elles peuvent être corrigées. Dans le cas du temps de transactionΓ la valeur temporelle est fournie par le systèmeΓl'utilisateur fournit toujours la valeur structurelle. Bien que celle dernière puisse être erronéeΓ sa modification est impossibleΓ l'utilisateur doit saisir la correction avec un nouveau temps de validitéΓla valeur erronée est conservée.

Les historiques observés selon le temps de transaction ont des caractérisques bien particulières [Sno95a] : les valeurs appartiennent toujours au passé et il s'agit d'historiques en escalier.

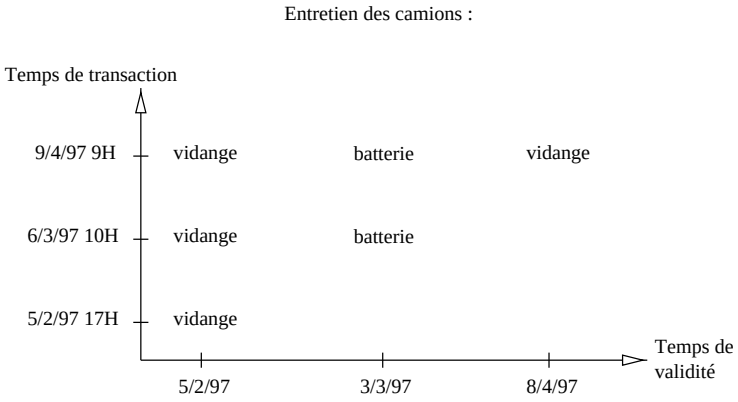
Le temps de transaction permet de mémoriser la connaissance que le SGBD avait du monde réel à un moment donné. Il permet également de retrouver les corrections qui ont



(a) Historique suivant le temps de validité



(b) Historique suivant le temps de transaction



(c) Historique bitemporel

FIG. 1.13 – *Les différentes dimensions temporelles*

été faites dans la base de données. Par exemple, dans la figure 1.14, on a enregistré le 1/5/97 à 9H que la réparation qui avait été faite le 3/3/97 n'est pas un changement de batterie mais une vidange.

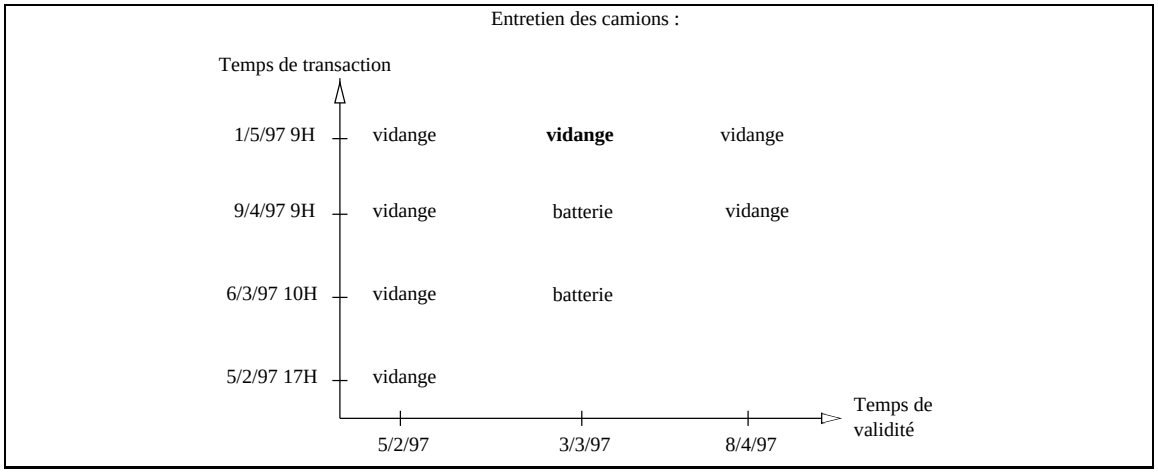


FIG. 1.14 – *Correction dans un historique bitemporel*

[SA85] propose une nommenclature des bases de données temporelles suivant les dimensions temporelles prises en compte :

- *Bases de données de reprise* (*rollback databases*) : bases de données ne supportant que le concept de temps de transaction, et qui permettent de suivre l'évolution des données au cours des différentes transactions.
- *Bases de données historiques* (*historical databases*) : bases de données ne gardant trace que de l'évolution selon le temps de validité.
- *Bases de données bitemporelles* (*bitemporal databases*) : bases de données qui supportent le temps de validité et le temps de transaction.

Le temps de validité et le temps de transaction ne sont pas les seules dimensions temporelles possibles, il est possible d'en définir d'autres. Dans les propositions de modèles d'historiques, il est aussi couramment admis un temps de l'utilisateur (*user-defined time* dans [JCE+94]), dont la sémantique est fixée par le domaine de l'application.

1.2.3 Modèles à valeurs

Dans un contexte relationnel, les historiques peuvent être définis soit au niveau des n-uplets, soit à celui des attributs. Le temps (de validité ou de transaction) peut être représenté par des instants, des intervalles ou des éléments temporels. L'utilisation d'éléments temporels permet de ne pas dupliquer de n-uplet lorsque une entité a la même

valeur pendant deux périodes de temps disjointes. Les historiques sont généralement interprétés comme étant en escalier. Lorsque le temps est exprimé à l'aide d'instantsΓcela exclut toute absence de valeur dans l'histoire des entités puisque la valeur estampillée à l'instant i1 estΓpar définitionΓvalide jusqu'à la valeur suivanteΓestampillée par i2. TSQL 2 propose aussi un type de relation correspondant à un historique discretΓle temps est alors représenté par un instant.

Lorsque le modèle associe les historiques aux n-upletsΓoption choisie par exemple dans [Gad93ΓNA87ΓNA93ΓSar93ΓSno94b]Γdes attributs particuliers (explicites ou implicites) permettent de gérer les temps de validité et/ou de transaction.

Dans cette approcheΓles relations sont maintenues en première forme normale et les acquis de cette propriété sont ainsi conservés aux niveaux conceptuel et logique. En contrepartieΓelle présente les défauts connus du modèle relationnel concernant notamment la finesse de structuration. Et l'introduction des attributs temporels augmente la redondance d'informations. Chaque fois que l'une des valeurs d'attribut est modifiéeΓun nouvel n-uplet doit être ajouté.

L'association des historiques aux attributsΓoption choisie par exemple dans [Tan93Γ GN93ΓCC93ΓGV85]Γpermet une modélisation plus fine de l'évolution des informations. Mais les relations ne sont plus en première forme normaleΓce qui a des conséquences aux niveaux logique et physique.

Nous illustrons ces deux méthodes (Cf. figure 1.15) en donnant la valeur de la relation des clients de notre entreprise-exemple Oplate (pour simplifierΓnous ne retenons que l'historique de leur adresseΓde leur numéro de téléphone et de leur directeur). Dans cette figureΓles intervalles dont la borne supérieure est notée "−" correspondent aux valeurs définies par la dernière mise à jour.

Dans la figure 1.15b les périodes de validité apparaissent sous forme d'intervalles (attribut Validité). Dans la modélisation non temporelleΓla clef de la relation est définie sur l'attribut Nom. Dans la modélisation temporelleΓil faut y ajouter l'attribut Validité.

L'examen de la figure 1.15 met en évidence la redondance d'information induite par l'association des historiques aux n-uplets et les différences de traitement en cas de modification. SupposonsΓpar exempleΓque M. Martin deviennent le nouveau directeur de l'Alimentation du maquis le 15 avril 1996.

Dans le premier cas (Cf. figure 1.15a)Γil faut :

- Modifier la période de validité du n-uplet le plus récent concernant ce client :
  ([1/4/93, −] devient [1/4/93, 14/4/96])

- Ajouter le n-uplet :
  <Alimentation du maquis, Propriano, 04 95 44 57 22, Martin, [15/4/96, −]>.

Nom	Adresse	Téléphone	Directeur	Validité
Supermarché du désert	Tataouine	162 243	Hiddid	[1/1/90Γ30/11/94]
Supermarché du désert	Tunis	325 478	Hiddid	[1/12/94Γ−]
Alimentation du maquis	Propriano	04 95 34 12 56	Dupond	[1/3/90Γ20/5/92]
Alimentation du maquis	Ajaccio	04 95 56 39 74	Dupond	[21/5/92Γ15/8/92]
Alimentation du maquis	Ajaccio	04 95 56 39 74	Maurizi	[1/9/92Γ31/3/93]
Alimentation du maquis	Propriano	04 95 44 57 22	Maurizi	[1/4/93Γ−]

(a) Historiques au niveau des n-uplets

Nom	Adresse	Téléphone	Directeur
Supermarché du désert [1/1/90Γ−]	Tataouine [1/1/90Γ30/11/94] Tunis [1/12/94Γ−]	162 243 [1/1/90Γ30/11/94] 325 478 [1/12/94Γ−]	Hiddid [1/1/90Γ−]
Alimentation du maquis [1/1/90Γ−]	Propriano [1/3/90Γ20/5/92] Ajaccio [21/5/92Γ31/3/93] Propriano [1/4/93Γ−]	04 95 34 12 56 [1/3/90Γ20/5/92] 04 95 56 39 74 [21/5/92Γ31/3/93] 04 95 44 57 22 [1/4/93Γ−]	Dupond [1/3/90Γ15/8/92] Maurizi [1/9/92Γ−]

(b) Historiques au niveau des attributs

FIG. 1.15 – *Niveau d'observation des historiques dans le modèle relationnel*

R1

Nom	Adresse	Téléphone	Validité
Supermarché du désert	Tataouine	162 243	[1/1/90Γ30/11/94]
Supermarché du désert	Tunis	325 478	[1/12/94Γ−]
Alimentation du maquis	Propriano	04 95 34 12 56	[1/3/90Γ20/5/92]
Alimentation du maquis	Ajaccio	04 95 56 39 74	[21/5/92Γ31/3/93]
Alimentation du maquis	Propriano	04 95 44 57 22	[1/4/93Γ−]

R2

Nom	Directeur	Validité
Supermarché du désert	Hiddid	[1/1/90Γ−]
Alimentation du maquis	Dupond	[1/3/90Γ15/8/92]
Alimentation du maquis	Maurizi	[1/9/92Γ−]

FIG. 1.16 – *Historiques au niveau des n-uplets : décomposition en deux relations*

Dans le deuxième cas (Cf. figure 1.15b)Γil ne faut intervenir que sur l'attribut Directeur associé à l'Alimentation du maquis en :

- Remplaçant <Maurizi, [1/9/92, −]> par <Maurizi, [1/9/92, 14/4/96]>
- Et en ajoutant <Martin, [15/4/96, −]>.

La figure 1.15a permet d'illustrer la notion d'évolution synchrone des attributs d'une relation. Par exempleΓiciΓles attributs adresse et téléphone évoluent de manière synchroneΓ mais l'attribut directeur est asynchrone par rapport à eux. De ce faitΓle changement du directeur implique la duplication de l'adresse et du numéro de téléphone dans un nouveau n-uplet. De plusΓl'information de la période de validité d'un attribut est répartie sur plusieurs n-upletsΓpar exemple la validité de l'adresse d'Ajaccio pour l'Alimentation du maquis. Cela peut compliquer l'expression et l'implantation de certaines requêtes.

Pour éviter ces problèmesΓon cherche alors à décomposer la relation considérée en sous-relations dont tous les attributs évoluent de manière synchrone. On définit à cet effet une relation d'équivalence sur un ensemble d'attributs : tous les attributs d'une classe évoluent de manière synchrone. Les classes définies par cette relation sont alors à la base de la décomposition (*classe d'équivalence synchrone* [NA93]).

Dans cet exempleΓon peut répartir les attributs variables en deux classes d'équivalence synchronesΓd'un part {Adresse, Téléphone} et d'autre part {Directeur}. Ceci conduit à décomposer la relation initiale en deux sous-relations R1 (Nom, Adresse, Téléphone, Validité) et R2 (Nom, Directeur, Validité) (Cf. figure 1.16).

La redondance d'informations deumeure lorsque le domaine de l'attribut validité est celui des instants ou des intervalles. Celle-ci peut être évitée par l'utilisation d'éléments temporels : dans la figure 1.15bΓles couples < Propriano, [1/1/90, 30/11/94] > et < Propriano, [1/4/93, −] > sont regroupés en un seul < Propriano, [1/1/90, 30/11/94] ∪ [1/4/93, −] >.

1.2.4 Modèles à objets

Comme dans le cas du relationnelΓles modèles à objets permettent d'associer les historiques aux objets [KS92ΓSC91ΓSN97] ou aux attributs [PEA+95ΓRS93]. OODAPLEX [WD93] permet indifféremment l'une ou l'autre possibilité.

Selon le modèleΓles attributs historiques sont décrits par des fonctions à domaine temporel (par exemple dans OODAPLEX [WD92ΓWD93]) ou par des séquences temporelles (par exemple [ADF+94]). La figure 1.17 illustre ces deux possibilités dans le cas de l'association des historiques aux attributs.

La figure 1.18 illustre l'association des historiques aux objets ou aux attributs. Lorsque les historiques sont associés aux objetsΓles techniques de modélisation couramment employées dans le domaine plus général des versions peuvent être utilisées [Fau89ΓSci91Γ Fau92ΓWD93ΓSci94]. Deux classes d'objets sont alors définies : la première associe à un

objet son historiqueΓc'est-à-dire la séquence de ses versions ordonnées dans le temps ; la deuxième classe décrit une vision instantanée de l'objetΓc'est-à-dire un état de l'objet à un instant donné.

L'évolution synchrone d'attributs peut être exprimée par l'introduction d'un niveau d'abstraction supplémentaire correspondant aux classes d'équivalence synchrones mises en évidence lors de l'analyse.

1.3 Langages

Les extensions attendues des systèmes concernent la possibilité d'identifier les entités et associations dont on veut pouvoir observer l'évolutionΓde spécifier les instants où leur changement est pertinent et de disposer d'opérateurs temporels permettant d'exprimer le plus naturellement possible des requêtes d'interrogationΓde création et de mise à jour en reliant faits et événements dans le temps.

Dans les modèles à objetsΓon exploite directement la puissance d'expression des langages de programmation (au travers des méthodes) et la possibilité de définition de bibliothèques de classes.

Dans les modèles à valeursΓla démarche générale consiste à étendre la définition des expressions en incluant les types temporelsΓen étendant les opérations usuelles de projectionΓsélection et produit et les opérations d'agrégation. Il est aussi nécessaire de rajouter des opérateurs particuliers liés à l'ordre chronologique entre les n-uplets.

1.3.1 Extension de l'algèbre relationnelle

Il existe de nombreuses extensions temporelles de l'algèbre relationnelle [LJ88ΓKaf93ΓDBS96]. Nous résumons ici les principaux opérateursΓen présentant la(les) adaptation(s) temporelle(s) généralement proposée(s). Pour celaΓnous nous basons sur les algèbres de HRDM [CC93]Γqui associe les historiques aux attributsΓet de TSQL 2 [Sno95b]Γqui associe les historiques aux n-uplets.

– *Projection temporelle.*

  Dans le cas de l'association des historiques aux attributsΓil s'agit d'une extension directe de la projection habituelle. Dans l'autre casΓla projection selon un attribut donné réalise une fusion (*coalesce*) du temps de validité (ou de transaction) associé à chaque valeur ; ce temps pouvant être réparti sur plusieurs n-uplets pour une même valeur.

  La requête "Pendant quelle période Mr Dupond a-t-il été directeur de l'alimentation du maquis" est un exemple de projection temporelle. Si le temps est associé aux attributsΓla réponse est le temps de validité associé à la valeur "Dupond" de l'attribut Directeur.

TYPE Client :

Nom : Client $\longrightarrow$ texte

Adresse : Client $\longrightarrow$ (date $\longrightarrow$ texte)

Téléphone : Client $\longrightarrow$ (date $\longrightarrow$ texte)

Directeur : Client $\longrightarrow$ (date $\longrightarrow$ texte)

(a) A l'aide de fonctions à domaines temporels

TYPE Client :

Nom : Client $\longrightarrow$ texte

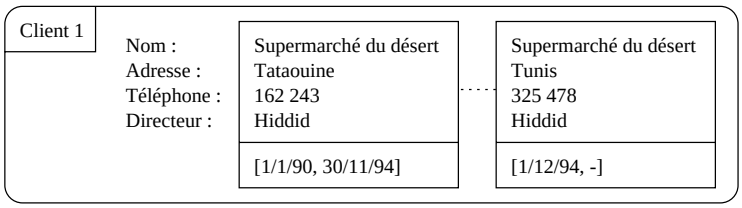
Adresse : [< texte, élément temporel>]

Téléphone : [< texte, élément temporel >]

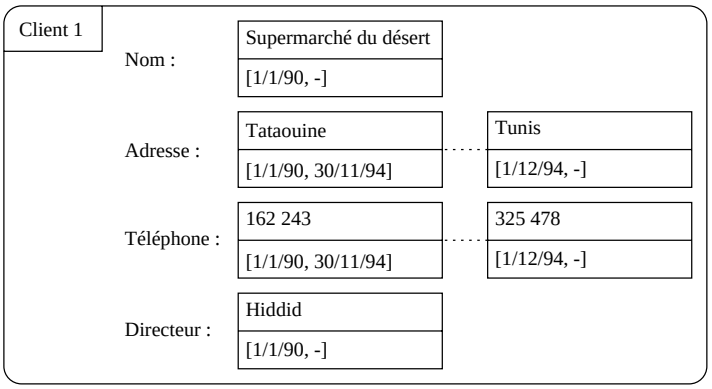
Directeur : [< texte, élément temporel >]

(b) A l'aide de séquences temporelles

FIG. 1.17 – *Différentes descriptions des historiques dans les modèles à objets*



(a) Historiques au niveau des objets



(b) Historiques au niveau des attributs

FIG. 1.18 – *Niveau d'observation des historiques dans les modèles à objets*

La réponse est [1/3/90Γ 15/8/92] (Cf. figure 1.15a). Si le temps est associé aux n-upletsΓla réponse est l'union des temps de validité associés aux n-uplets dont la valeur de l'attribut Directeur est "Dupond". IciΓla réponse est l'union des intervalles [1/3/90Γ 20/5/92] et [21/5/92Γ15/8/92]Γsoit l'intervalle [1/3/90Γ15/8/92] (Cf. figure 1.15b).

– *Sélection temporelle.*

  Elle permet de retenir toute l'histoire de n-uplets vérifiant une certaine propriété à certains moments dans le temps. Cela correspond à la question : "Type et historique du prix des bouteilles dont le prix a dépassé les 5 francs avant le 31/12/1996".

– *Restriction temporelle.*

  Elle permet de ne retenir dans une relation que la restriction des n-uplets à une période donnéeΓpar exempleΓ"Quelle est l'histoire du client "Supermarché du désert" pendant la période [1/1/90Γ31/12/99] ?"

– *Sélection et Restriction.*

  Il s'agit ici de combiner une sélection temporelle avec une restriction temporelle. Cela permet de restreindre l'histoire des n-uplets sélectionnés aux instants où ils vérifient la propriété spécifiée. Par exemple la réponse à la requête "Quelle est l'historique des bouteilles "PlusSoif" lorsque celles-ci coû moins de 3 francs ?" sera une relation déduite de la relation Bouteille en restreignant tous les attributs aux instants correspondant à la propriété demandée.

– *Produit cartésien temporel.*

  Le produit cartésien est étendu selon deux options : considérer que la période de validité des n-uplets du résultat est l'union ou l'intersection des périodes de validité des n-uplets argumentsΓcomme l'illustre la figure 1.19. Selon l'option choisieΓles relations obtenues sont temporellement homogènes ou non.

– *Théta produit et produit naturel.*

  Dans les deux casΓc'est une combinaison du produit cartésienΓd'une sélection et d'une restriction temporelle. On ne retient que les n-uplets vérifiant au moins une fois la propriété donnée (implicite pour le produit naturel) et on les restreint aux périodes de validité où cette propriété est vérifiée.

– *Opérateurs ensemblistes.*

  Ils sont redéfinis de manière à traiter correctement les cas de chevauchement des périodes de validité d'attributs intervenant dans l'opération : par exempleΓdans le cas d'une unionΓplutôt que de produire deux n-uplets ne se distinguant que par la période de validitéΓon réalise une "fusion"Γqui produit un seul n-uplet dont la période de validité résulte de l'union des périodes de validité.

R1	A1	Validité
	AA	i_1
	AB	i_3
	AC	i_4
	AD	i_7

R2	A2	Validité
	ZA	i_3
	ZB	i_4
	ZC	i_5
	ZD	i_6

(a) Exemple de deux relations R1 et R2

R1 $\times_{t,\cap}$ R2	A1	A2	Validité
	AB	ZA	i_3
	AC	ZB	i_4

(b) Produit cartésien temporel avec la sémantique de l'intersection, noté $\times_{t,\cap}$

R1 $\times_{t,\cup}$ R2	A1	A2	Validité
	AA		i_1
	AB	ZA	i_3
	AC	ZB	i_4
		ZC	i_5
		ZD	i_6
	AD		i_7

(c) Produit cartésien temporel avec la sémantique de l'union, noté $\times_{t,\cup}$

FIG. 1.19 – *Extension du produit naturel au contexte temporel*

1.3.2 Typologie de requêtes temporelles

Les critères de classification que nous avons retenus pour choisir les requêtes (Cf. annexe A.1.6) portent sur la nature des trois composants de base d'une requête temporelle (dans [EPP94]Γla typologie proposée est fondée uniquement sur l'analyse de la nature de la sélection et de la projection).

- La sélection exprime les conditions que doivent vérifier les objets pour contribuer au résultat. Nous disons qu'une sélection est :

  - TemporelleΓlorsque l'expression de sélection fait référence au temps : *...être client pendant la période [1/1994, 7/1994].*
  - Non temporelle dans le cas contraire : *...être situé à Tataouine.*

- La projection spécifie le schéma du résultat attendu. Nous disons qu'une projection est :

  - Temporelle : lorsque le résultat contient des informations temporelles :
    - Sous forme d'un historique : *Historique de la production des bouteilles.*
    - Ou sous forme de valeurs temporelles : *Durées des périodes entre deux révisions d'un camion.*
  - Non temporelle lorsque le résultat ne contient pas d'informations temporelles : *Nom des clients.*

- La définition du domaine temporel du résultat lorsque celui-ci est un historique. Cet aspect de la requête peut être exprimé :

  - Explicitement il s'agit alors d'une *restriction temporelle* c'est-à-dire d'une restriction du domaine temporel de l' historique : *Pour chaque client, donner l'historique de ses directeurs, restreint à la période [1990, 1995].*
  - Implicitement le domaine temporel de l'historique étant déduit des instants pendant lesquels une condition de sélection est vérifiée : *Pour chaque type de bouteille, donner l'historique de sa production, pendant qu'il est vendu plus de 5 francs.*

    Nous parlons de domaine temporel *déduit d'une synchronisation* lorsqu'il est défini par la mise en correspondance de deux historiques : les instants sélectionnés sont ceux des instantanés vérifiant simultanément une même condition.
  - Par une combinaison des deux : *Pour chaque type de bouteille, donner l'historique de sa production, restreint à la période [1990, 1992], pendant qu'il est vendu plus de 5 francs.*

1.3.3 Expression de requêtes dans différents langages

Afin de comparer l'expression de requêtes temporelles dans différents langages nous présentons ici quelques-unes des requêtes relatives à notre exemple Oplate.

Nous avons retenu des langages de requêtes non-temporels (SQL pour le relationnel et OQL pour l'objet) et temporels (TempSQL TSQL 2 pour le relationnel et TOOSQL pour l'objet).

L'annexe A comporte la description de l'application la modélisation des entités dans les différents modèles retenus ainsi que l'expression de l'ensemble des requêtes dans chaque langage.

Nous en avons extrait trois afin d'illustrer les trois composantes de base des requêtes temporelles (sélection/projection et définition du domaine temporel).

- **R.3** *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

  - **SQL** :

```
PrixPeriode =
{ Cette requête permet de construire l'historique des prix des bouteilles dans
[1/1/1990, 31/12/1996]. }
SELECT *
{ Ensemble des prix des types de bouteilles modifiés dans la période. }
FROM BouteillePrix
WHERE Validite > '1/1/1990' AND Validite ≤ '1/12/1996'
UNION
SELECT '1/1/1990', Type, Prix
{ Pour chaque type de bouteille, détermination du prix valide au 1/1/1990 (lors-
qu'il est défini à cette date là). }
FROM BouteillePrix P1
WHERE Validite = (SELECT MAX(Validite)
                  FROM BouteillePrix P2
                  WHERE Validite ≤ '1/1/1990'
                  AND P2.Type = P1.Type)
UNION
SELECT '1/1/1990', Type, NULL
{ Pour chaque type de bouteille dont le prix n'est pas défini au 1/1/1990,
construction d'un n-uplet avec la valeur absente. }
FROM BouteillePrix P1
WHERE Type NOT IN (SELECT Type
                   FROM BouteillePrix
                   WHERE Validite < '1/1/1990'

Requête finale :
SELECT *
FROM PrixPeriode
WHERE Type NOT IN (SELECT Type
                   FROM PrixPeriode
                   WHERE Prix IS NULL)
```

  La requête finale retient le type et l'historique des prix pour les bouteilles dont le prix n'a jamais été absent (donc jamais à NULL) pendant la période donnée.

– **TSQL 2 :**

SELECT B.Type, B.Prix VALID INTERSECT (PERIOD '[1/1/1990, 1/12/1996]')
{ *La clause VALID INTERSECT effectue la restriction temporelle à la période donnée de la relation résultat de la requête.* }
FROM BouteillePrix B
WHERE VALID(B) CONTAINS PERIOD '[1/1/1990, 1/12/1996]'
{ *La clause WHERE introduit ici une sélection temporelle.* }

– **TempSQL :**

SELECT Type, Prix
WHILE [1/1/1990, 1/12/1996]
{ *La clause WHILE introduit la restriction temporelle, à la période donnée, de la relation résultat.* }
FROM Bouteille
WHERE $[\![$Prix $]\!] \subseteq$ [1/1/1990, 1/12/1996]
{ *L'expression* $[\![Prix]\!]$ *constuit le domaine temporel de l'attribut Prix du n-uplet traité.* }
{ *La condition permet de ne retenir que les n-uplets dont le prix est défini pendant tout l'intervalle.* }

– **OQL :**

DEFINE PrixPériode AS
{ *Cette requête permet de construire l'historique des prix des bouteilles dans [1/1/1990, 31/12/1996].*
Le type du résultat est { < *string, [<date, real>]* > }. }
SELECT TUPLE(Type : B.Type,
HistPrix: SELECT P1 FROM P1 IN B.Prix
{ *Ensemble des prix de bouteille modifiés dans la période.* }
WHERE P1.Validite $\geq$ '1/1/1990'
AND P1.Validite $\leq$ '1/12/1996'
UNION
SELECT TUPLE(Validite: '1/1/1990',
Prix: P2.Prix)
{ *Pour chaque type de bouteille, détermination du prix valide au 1/1/1990 (lorsqu'il est défini à cette date là).* }
FROM P2 in B.Prix
WHERE P2.Validite = MAX(SELECT P3.Validite
FROM P3 in B.Prix
WHERE P3.Validite < '1/1/1990'))
FROM B IN LesBouteilles

Requête finale :
{ *type du résultat :* { < *string, [<date, real>]* > } }
SELECT H
FROM H IN PrixPériode
WHERE MIN(SELECT P.Validite FROM P IN H.HistPrix) = '1/1/1990'
AND FOR ALL H IN PrixPériode : H.Valeur ≠ nil

La requête finale permet de ne retenir que les types de bouteilles dont le prix est défini dès le premier instant de la périodeΓet pendant toute la période.

- **TOOSQL** :

SELECT B.Type, B.Prix.History
FROM B:Bouteille
TIME-SLICE [1/1/1990..1/12/1996]
{ *La clause TIME-SLICE restreint le résultat aux n-uplets valides pendant la période donnée* }
WHERE B.Prix.Duration(vt).Sum = Duration([1/1/1990..1/12/1996])
{ *La méthode Duration(vt) appliquée à un attribut historique construit les durées de chaque intervalle de la séquence temporelle associée.* }
{ *La méthode Sum effectue la somme de ces durées* }

Nous prenons ici pour hypothèse que la restriction temporelle introduite par la clause TIME-SLICE est effectuée avant l'évaluation de la condition de la clause WHERE.

- **R.7** *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

- **SQL** :

PrixProduction =
{ *Cette requête intermédiaire réunit les historiques de production et de prix (sans restreindre à un domaine temporel particulier).* }
SELECT O.Validité, O.Type, O.Production, I.Prix
FROM BouteilleProd O, BouteillePrix I
WHERE O.Type = I.Type
{ *La disjonction ci-dessous permet de déterminer le prix d'une bouteille à chaque instant où une valeur de sa production est donnée. Les n-uplets retenus sont ceux pour lesquels le prix correspond à l'instant de la production, soit* t_1, *que le prix ait été changé à cet instant où à un instant précédent, soit* t_2 *(sans autre modification de prix entre* t_1 *et* t_2*).* }

```
AND (I.Validité = O.Validité
      OR
      I.Validité < O.Validité
      AND NOT EXISTS (SELECT *
                      FROM BouteillePrix II
                      WHERE II.Type = I.Type
                      AND II.Validité > I.Validité
                      AND II.Validité ≤ O.Validité))
```

Requête finale :

```
SELECT SUM(Production)
FROM PrixProduction
WHERE Type = 'PlusSoif' AND Prix > 5
```

Cette requête restreint la relation au type et au prix considéréΓet calcule la somme des bouteilles produites.

- **TSQL 2** :

```
SELECT SUM(O.Production)
FROM BouteillePrix P, BouteilleProduction O
WHERE O.Type = P.Type AND P.Prix > 5
```

TSQL 2 est particulièrement bien adapté à l'expression de requêtes dans lesquelles on combine des instantanés issus d'historiques différents et définis aux mêmes instants.

- **TempSQL** :

```
{ type du résultat: historique de productions }
SELECT Production
WHILE [[Prix > 5 ]]
FROM Bouteille
```

Pour donner la quantité globale produite du type de ces bouteillesΓil est nécessaire de disposer d'un opérateur de projection d'un historique selon ses valeurs structurelles afin d'appliquer dessus des opérations arithmétiques. Un tel opérateur n'est pas défini dans TempSQL.

- **OQL :**

```
SUM(SELECT Prod.Valeur
    FROM B IN LesBouteilles, Prod IN B.Production, Prx in B.Prix
    WHERE B.Type = 'PlusSoif'
    AND Prx.Valeur > 5
    { La disjonction ci-dessous permet de déterminer le prix d'une bouteille
    à chaque instant où une valeur de sa production est donnée. Les n-uplets
    retenus sont ceux pour lesquels le prix correspond à l'instant de la pro-
    duction, soit t1, que le prix ait été changé à cet instant où à un instant
    précédent, soit t2 (sans autre modification de prix entre t1 et t2). }
    AND (Prx.Validite = Prod.Validite
        OR
        Prx.Validite < Prod.Validite
        AND NOT EXISTS Prx2 IN B.Prix :
                Prx2.Validité > Prx.Validité
                AND Prx2.Validité < Prod.Validité))
```

- **TOOSQL :**

```
SELECT B.Production.Sum
FROM B:Bouteille
WHERE B.Type = 'PlusSoif'
AND B.Prix > 5
```

- **R.9** *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

- **SQL :**

```
SELECT C2.Adresse
FROM ClientCoordonnees C1, ClientCoordonnees C2
WHERE C1.Nom = C2.Nom
AND C1.Nom = 'Alimentation du maquis'
AND C1.Adresse = 'Propriano'
AND C1.Validite < C2.Validite
AND NOT EXISTS (SELECT *
                FROM ClientCoordonnees C3
                WHERE C3.Nom = C1.Nom
                AND C3.Validite > C1.Validite
                AND C3.Validite < C2.Validite)
```

Le requête principale fait le lien entre l'adresse à Propriano et les adresses suivantes. La sous-requête permet de sélectionner le n-uplet C2 correspondant à l'adresse où le client a déménagé juste après avoir quitté Propriano.

- **TSQL 2 :**

```
SELECT SNAPSHOT C1.Adresse
FROM ClientCoordonnées C1
WHERE C1.Nom = 'Alimentation du maquis'
AND EXISTS (SELECT *
            FROM C2 in ClientCoordonnées
            WHERE C2.Nom = C1.Nom AND C2.Adresse = 'Propriano'
            AND VALID(C1) MEETS VALID(C2))
```

La clause SELECT SNAPSHOT exprime la projection de la relation résultat selon les valeurs structurelles. Le résultat est une relation instantanée.

- **TempSQL :**

Cette requêteΓutilisant un raisonnement basé sur la succession dans le tempsΓne peut pas être exprimée avec les opérateurs définis dans [GN93].

- **OQL :**

```
SELECT CC2.Valeur.Adresse
FROM C IN LesClients, CC1 IN C.Coordonnées, CC2 IN C.Coordonnées
WHERE C.Nom = 'Alimentation du maquis'
AND CC1.Valeur.Adresse = 'Propriano'
AND CC2.Validite > CC1.Validite
AND NOT EXISTS CC3 IN C.Coordonnées :
        (CC3.Validité > CC1.Validité AND CC3.Validité < CC2.Validité)
```

On retrouve dans cette requête en OQL les mêmes mécanismes que dans ceux mis en oeuvre dans son expression en SQL.

- **TOOSQL :**

```
SELECT C.Adresse
FROM C:Client
WHERE C.Nom = 'Alimentation du maquis'
WHEN C.Adresse.vt.Follows (SELECT CC.Adresse.vt
                           FROM CC:Client
                           WHERE CC.Nom = C.Nom AND CC.Adresse = 'Propriano')
                           { Cette première condition permet de ne sélection-
                           ner que les adresses après que le client ait quitté
                           Propriano. }
AND NOT EXISTS (SELECT *
                FROM CCC:CLient
                WHERE CC.Nom = CCC.Nom
                WHEN CCC.Adresse.vt.Follows (C.Adresse.vt)
                AND CC.Adresse.vt.Follows (CCC.Adresse.vt))
```

{ *Cette seconde condition permet de ne retenir que les adresses correspondant à l'adresse où à déménager le client, juste après avoir quitté Propriano.* }

A la vue de ces requêtes, il est possible de faire quelques remarques générales. Sur les langages non-temporels :

- Le seul type temporel de base est le type Date. Pour manipuler des intervalles ou des éléments temporels, il faut les représenter explicitement ; de même les opérations (relations d'ordre, opérations ensemblistes, etc.) sont programmées dans chaque requête ce qui en accroît la complexité.

- Le choix de la représentation des associations temporelles (à base d'instants, d'intervalles ou d'élements temporels) est laissé à la charge des applications. Ce choix étant fixé, si certaines requêtes nécessitent d'en changer, les opérations qui assurent le passage d'une représentation à une autre sont encore à la charge des applications.

- Les modèles à objet permettent de mieux représenter les valeurs et associations temporelles complexes et les opérations associées. Les structures de données alors obtenues nécessitent l'usage d'expressions de navigation, ce qui rend les requêtes plus complexes.

Les langages temporels fournissent des opérations sur les valeurs et associations temporelles mais :

- Le choix de la représentation des associations est imposé, par exemple à base d'intervalles dans TOOSQL et à base d'élements temporels dans TempSQL. Le changement de représentation est impossible (sauf si l'application le prend à sa charge en utilisant des relations (ou classes) non temporelles !).

- Les modèles ne fournissent aucun mécanisme d'interprétation des valeurs structurelles non saisies. Par exemple, dans le cas d'un phénomène physique, il est courant d'interpoler une valeur entre deux mesures.

De manière générale, le choix de représentation des valeurs et expressions temporelles influe sur l'expression des requêtes. Fixer ces représentations facilite l'écriture de certaines requêtes mais rend l'écriture des autres plus complexe.

Les raisonnements classiquement utilisés dans les langages de requêtes sont basés sur des notions ensemblistes, ce qui ne convient pas lorsqu'on veut exploiter la succession dans le temps.

1.3.4 Cas du bitemporel

Pour prendre en compte le bitemporelΓil faut encore étendre les opérations de l'algèbre. Dans le cas de TSQL 2Γtemps de validité et temps de transaction sont gérés exactement de la même manière. AussiΓil s'agit des mêmes opérateurs : TRANSACTION(x) permet de manipuler le temps de transaction d'un n-upletΓcomme VALID(x) permet de le faire pour le temps de validité.

Comme la complexité d'expression des requêtes est encore accrue dans le cas du bitemporelΓd'autres propositionsΓcomme TOOSQLΓoffrent des opérations spécifiques au temps de transaction. Nous avons vu que celui-ci est utile pour tracer des erreursΓse replacer dans des conditions du passéΓetc. Pour celaΓTOOSQLΓintroduit deux opérateurs :

- Rollback permet de spécifier un état passé de la base de données sur lequel la requête est évaluée.
- Without Correction permet de ne pas prendre en compte les corrections faites dans les historiques.

Exemples de requêtes portant sur le temps de validité (en rapport avec l'exemple de OplateΓet en considérant que les relations sont bitemporelles) :

- Pour chaque camionΓdonner son numéro et l'historique de ses visites d'entretien (nature et coût)Γtelles qu'elles étaient enregistrées dans la base le 1er janvier 1995Γlorsqu'il ne s'agissait pas de vidanges.
- Indiquer le nombre moyen de corrections effectuées dans la base sur les données relatives aux visites d'entretien des camions.

1.3.5 Mises à jour

Les opérations classiques de mise à jour (InsertΓ Update et Delete) sont adaptées au contexte temporel.

Dans le cas d'historiques suivant le temps de validitéΓl'utilisateur doit fournirΓoutre éventuellement une valeur structurelleΓune valeur temporelle datant la mise à jour. Dans le cas du temps de transactionΓcette estampille est automatiquement fournie par le système.

Nous illustrons ces opérations de mise à jourΓpour le temps de validitéΓdans le cadre du langage TSQL 2 :

- Insert : permet l'insertion de nouveaux n-uplets dans la base. Dans le cas où le domaine temporel du temps de validité associé au nouveau n-uplet recoupe le temps de validité d'un n-uplet de même valeur alors deux cas sont possibles :
  - Le système retient un seul couple et fusionne (*coalesce*) les deux temps de validité.

- Le système refuse l'insertion si une telle contrainte a été explicitement définie par l'utilisateur.

Exemple :

```
INSERT INTO CamionEntretiens
VALUES ('735 AOC 38', 'batterie', '800')
VALID TIMESTAMP '3/3/1997'
```

- Update. La mise à jour des attributs non-temporels se fait comme en SQL 2 : les valeurs d'un ou plusieurs attributs du n-uplet sont modifiéesΓsans que le temps de validité en soit affecté.
  Exemple :

```
UPDATE CamionEntretiens
SET Nature TO 'vidange'
WHERE Immat = '735 AOC 38' AND Nature = 'batterie'
```

  Le temps de validité peut aussi être modifié :

```
UPDATE CamionEntretiens
SET Nature TO 'vidange' VALID TIMESTAMP '3/4/1997'
WHERE Immat = '735 AOC 38' AND Nature = 'batterie'
```

- Delete. Cela permet de supprimer des n-uplets d'une relation en annulant des périodes de validité. Deux cas sont possibles suivant la nature de la relation :

  - valid state (la période de validité associée est un intervalle) : les tuples concernés par la condition donnée verront leur période de validité amputée de la période donnée. Dans le cas où la période résultante est videΓle n-uplet est effectivement supprimé de la relation.
  - valid event (la période de validité associée est un instant) : les tuples concernés par la condition donnée et dont l'estampille correspond à la valeur donnée sont supprimés de la relation.

  Exemple :

```
DELETE FROM CamionEntretiens
WHERE Immat = '735 AOC 38' AND Nature = 'vidange'
VALID TIMESTAMP '3/4/1997'
```

1.4 Architectures et prototypes

1.4.1 Architectures classiques

Il existe de nombreuses propositions d'extensions temporelles pour des SGBDΓparfois complétées par la réalisation de prototypes. Parmi tous ces travauxΓ[VGS96] dégage trois principaux types d'architecture :

- L'extensibilité propre aux SGBD à objets rend possible l'implantation de modèles de données temporels au dessus d'un SGBD existant sans avoir à intervenir sur son noyau. Par exempleΓles modèles TF-ORM et TOM ont été implantés avec le SGBD O_2 au moyen de bibliothèques de classes implantant les types temporels (instantsΓintervallesΓ etc.)Γles séquences temporelles (suites de couples <instantΓvaleur>) et éventuellement les opérateurs sur des collections d'objets temporels.

- D'autres modèles temporels sont implantés en rajoutant une interface entre les applications et un SGBD hôte (par exempleΓ[Böh94ΓBBJS97]). Cette approche est nécessairement utilisée pour toutes les extensions temporelles de modèles de données relationnels n'offrant pas la possibilité de définir de nouveaux types abstraits.

- Une troisième optionΓpeu explorée à notre connaissanceΓconsiste à intégrer les fonctionnalités temporelles dans le noyau du SGBD. AinsiΓla structure de l'extension temporelle est transparente aux applications et les performances du SGBD sont conservées.

[VGS96] analyse ces diverses architectures selon des critères comme la complétude des fonctionnalités temporellesΓla compatibilité avec une utilisation non-temporelleΓla facilité d'implantationΓles performancesΓetc.

Ainsi les propositions adoptant la première option sont facilement réalisable tandis que leur facilité d'utilisation est discutable. Dans le deuxième casΓla facilité d'utilisation est améliorée mais les performances sont diminuées car l'extension temporelle est une couche logicielle intermédiaire entre les applications et le SGBD. EnfinΓl'intégration dans le noyau du SGBD est sans doute la solution la plus performanteΓmais elle est peu utilisée car elle nécessite en général la complicité du constructeur du SGBD. De plusΓune telle solution n'est guère portable.

1.4.2 Prototypes : récapitulatif

La figure 1.20 regroupe l'ensemble des propositions présentée dans ce documentΓque ce soit des produits commerciauxΓdes prototypes ou des articles de recherches. Ne sont référencées ici que les propositions globales définissant un modèle de données et/ou un langage de requête ; les travaux spécifiques sont cités lorsqu'il en est question.

Nom	Nature [a]	Type	Modèle	Référence(s)
SQL 2	MDΓLR non temp.	Standard	Relationnel	[Ins92ΓCO93]
Oracle	SGBD	Industrie	Relationnel	[Ora92]
TempSQL	MDΓLR	Recherche	Relationnel	[GN93]
TSQL 2	MTΓMDΓLR	Recherche	Relationnel	[TSQΓSno95b]
[DBS96]	MDΓLR	Recherche	Relationnel	[DBS96]
Temp. modules	MTΓMDΓLR	Recherche	Relationnel	[WJS93ΓWJS95]
TFDL	MDΓLR	Recherche	Entités	[SK94]
MCO	Conception	Recherche	Objets	[Cas93]
ODMG	MDΓLR non temp.	Standard	Objets	[ADF$^+$94]
O_2	SGBD	Industrie	Objets	[Tec95]
OSAM*/T	MDΓLR	Recherche	Objets	[SC91]
TMAD	MDΓLR	Recherche	Objets	[KS92]
TIGUKAT	MDΓLR	Recherche	Objets	[GO93ΓOPS$^+$95]
TF-ORM	MDΓLR	Recherche	Objets	[EPP93ΓPEA$^+$95]
OODAPLEX	MTΓMDΓLR	Recherche	Objets	[WD93]
TOOSQL	MDΓLR	Recherche	Objets	[RS93]
Time Sequences	MDΓLR	Recherche	Indépendant	[SS93]

[a] MT : Modèle du temps, MD : Modèle de données temporel, LR : Langage de requêtes temporel

FIG. 1.20 – *Propositions étudiées*

Remarques concernant le choix des propositions :

- SQL 2 : Nous avons choisi SQL 2 plutôt que SQL 3 pour plusieurs raisons. Tout d'abordΓ c'est SQL 2 qui est implanté par la plupart des SGBD relationnels du commerce. De plus le projet SQL/Temporal (aspects temporels dans SQL 3) n'a commencé officiellement qu'en janvier 1996 (communiqué de presse de l'Accredited Standards Committee* X3)Γet n'est pas encore finalisé ; celui-ci tend à se rapprocher fortement de TSQL 2.
- ODMG : Nous faisons référence ici à la version 1 de la spécification. L'ODMG a annoncé le 28 juillet 1997 la version 2Γmais celle-ci ne fait pas de nouvelle proposition concernant les aspects temporels (Cf. serveur WWW de l'ODMGΓhttp://www.odmg.org/).

Pour chacune de ces propositionsΓnous avons résumé ses principales caractéristiques suivant les points abordés pendant ce chapitre (Cf. figure 1.21) : types temporelsΓmulti-granularité (et extensibilité des unités)Γtypes historiques et mode d'interrogation.

Proposition	**Types temporels**	**Multigranularité**	**Types historiques**	**Interrogation**
TempSQL	instantΓintervalleΓ élé. temporel	non	un seul typeΓ validité uniquement	langage temporel
TSQL 2	instantΓintervalleΓdurée pas d'élé. temporel	ouiΓextensible	deux types (escalierΓdiscret) transaction et/ou validité	langage et algèbre relationnels
Temp. modules	X [a]	ouiΓextensible	X	algèbre temporelle
OSAM*/T	instant	oui	un seul typeΓ validité uniquement	langage temporel
TMAD	date (instant)Γ intervalle	ouiΓprédéfini	un seul typeΓ validité uniquement	langage temporel
TIGUKAT	instantΓintervalleΓdurée pas d'élé. temporel	ouiΓextensible		algèbre et langage temporels
TF-ORM	instantΓintervalleΓdurée pas d'élé. temporel	ouiΓprédéfini	un seul typeΓ transaction et/ou validité	logique temporelle
OODAPLEX	instantΓ ensemble d'instants	ouiΓextensible	un seul typeΓ n dimensions	langage temporel
TOOSQL	date (instant)	ouiΓprédéfini	un seul typeΓ transaction et/ou validité	langage temporel
Time sequences	X	X	trois types (discretΓ escalierΓinterpolé)	langage temporel

FIG. 1.21 – *Caractéristiques du modèle du temps de diverses propositions*

[a] X signifie que ce point n'est pas abordé dans les articles de notre bibliographie relatifs à la proposition en question

1.5 Vers un serveur temporel

L'analyse d'une quinzaine d'implantations de SGBD temporels fournie dans [Böh95] souligne d'une part que la plupart sont fondées sur l'approche relationnelle et d'autre part elles offrent rarement les moyens d'étendre l'ensemble des unités de temps ou des calendriers disponibles.

Notre objectif est d'étudier la faisabilité d'un serveur temporel par la réalisation d'un prototype au dessus du SGBD O_2.

Nous présentons maintenant le modèle TEMPOS qui consiste à offrir des extensions temporelles pour un SGBD à objets. Il s'agit pour cela d'intégrer de nombreux résultats existants dans le cadre des modèles à objets.

Notre approche se caractérise en particulier par les objectifs suivants :

- Offrir un ensemble de types temporels de base permettant aux concepteurs d'applications d'exprimer les valeurs temporelles selon le calendrier de leur choix et de manipuler les unités de temps qu'ils désirent.

- Offrir un constructeur de types dédiés à la notion d'historiques et en particulier permettre de raisonner sur les historiques en faisant abstraction de leurs diverses représentations tant externes qu'internes.

- Offrir une classification des modalités de saisie (mises à jour) des informations correspondant aux diverses situations d'observation de phénomènes continus ou discrets.

Chapitre 2

Tempos : Modèle du temps

Sur la base des travaux de [WJS95]Γnous adoptons un modèle du temps numériqueΓ linéaire et discretΓapprochant le temps de manière granulaire par le biais d'un ensemble d'*unités d'observation du temps*. Ce modèle est choisi en particulier pour permettre de dater les instantanés composant les historiques par des valeurs numériques repérant des instants sur l'échelle du temps avec un degré de précision fixé a prioriΓmais variable selon le phénomène observé ou selon l'application. Par exempleΠl'évolution du chiffre d'affaires de notre entreprise Oplate sera observée sur une base mensuelle (sans chercher à affiner l'observation au niveau du jour) alors que la production de bouteilles le sera sur une base journalière et que le débit de la source le sera toutes les 6 heures.

AinsiΓdans ce modèleΠla notion d'unité d'observation du temps satisfait à plusieurs besoins : définir le niveau de granularité auquel on veut observer l'évolution des informations ; permettreΓsous certaines conditionsΓde définir la notion de durée ; représenter les valeurs temporelles de diverses manièresΓpour la représentation interne ou pour leur forme externeΓen particulier en intégrant le découpage usuel des calendriers en périodes comme par exemple les mois et les années.

2.1 Unités d'observation du temps

HabituellementΠla désignation d'un instant (un point précis du temps) se fait à l'aide de repères temporels qui peuvent varier d'un calendrier à un autre. Un calendrier représente un ensemble de repères qui sont traditionnellement utilisés dans une culture donnéeΓ dans un certain domaine d'applicationΓetc.

Par exempleΓdans le calendrier grégorien que nous utilisons habituellementΓnous exprimons une date à l'aide d'annéesΓde mois et de jours : le 14 Juillet 1997.

ParallèlementàcelaΓet quel que soit le domaine d'applicationΓnous disposons d'unités

qui permettent de donner des mesures (que ce soit des longueurs ou des durées). Par exemple, 2 jours définit une durée. Les mois, par contre, ne peuvent être considérés comme une mesure du temps car ils ne sont pas réguliers, et "1 mois" ne désigne pas un espace de temps de longueur fixe.

Cependant, dans la vie courante, il y a souvent fusion des deux notions (repère calendaire et unité de mesure) et l'on utilise indifférement tous ces repères aussi bien pour situer un instant que pour donner une durée. Il est tout à fait courant de donner des durées exprimées avec des années : par exemple, la durée de validité d'un contrôle technique est de 1 an. Exprimer des durées comme cela implique une certaine imprécision. Par exemple, dans le cadre du contrôle, que la durée en jours soit de 365 ou 366 est sans importance.

Dans ce contexte, nous présentons un modèle du temps, inspiré de [WJS95], basé sur un système extensible d'unités temporelles. De cette façon, la manipulation des valeurs temporelles (telles que instants, durées, etc) est indépendante des unités utilisées, qui peuvent être définies à loisir selon les besoins des applications.

Par ailleurs, ce modèle propose une seule notion d'unité d'observation du temps qui regroupe à la fois les unités de temps (permettant d'exprimer des mesures) et les repères temporels (permettant d'exprimer des dates).

2.1.1 Partition du temps

L'espace continu du temps étant vu comme une droite de réels, le temps est discrétisé par une partition de cette droite en une suite d'intervalles consécutifs disjoints (Cf. figure 2.1).

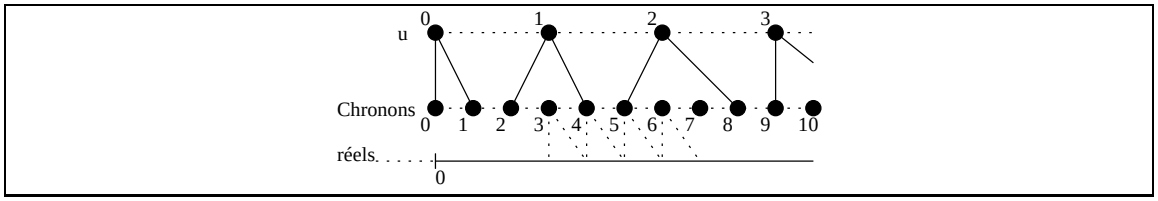


FIG. 2.1 – *Une unité d'observation du temps définit une partition*

Chaque intervalle est vu comme un *grain* non décomposable, et tout instant de la droite de réels est approximé par le grain qui le contient. Les grains sont numérotés par des entiers, l'ordre entre les entiers correspondant à la succession dans le temps et la distance entre deux entiers à la notion de durée. Nous choisissons ici de borner le temps par un instant origine et de numéroter les grains par des entiers naturels à partir de 0 (Cf. figure 2.1), de telle sorte que le numéro d'un grain représente la distance le séparant de l'origine.

Une alternative est de numéroter les grains par des entiers relatifs. Les grains de numéros positifs (respectivement négatifs) correspondent alors au futur (respectivement

au passé) par rapport à l'instant d'origine de numéro 0. Il est de plus possible de borner le temps dans le futur. Ceci est utile pour le contrôle d'intégrité dans le cas d'applications où le domaine temporel est fini. Nous ne le faisons pas ici pour ne pas alourdir l'exposé lors de la définition des fonctions sur les types temporels.

Une partition est définie par une fonction u de N dans 2^N dont les propriétés sont données par la figure 2.2.

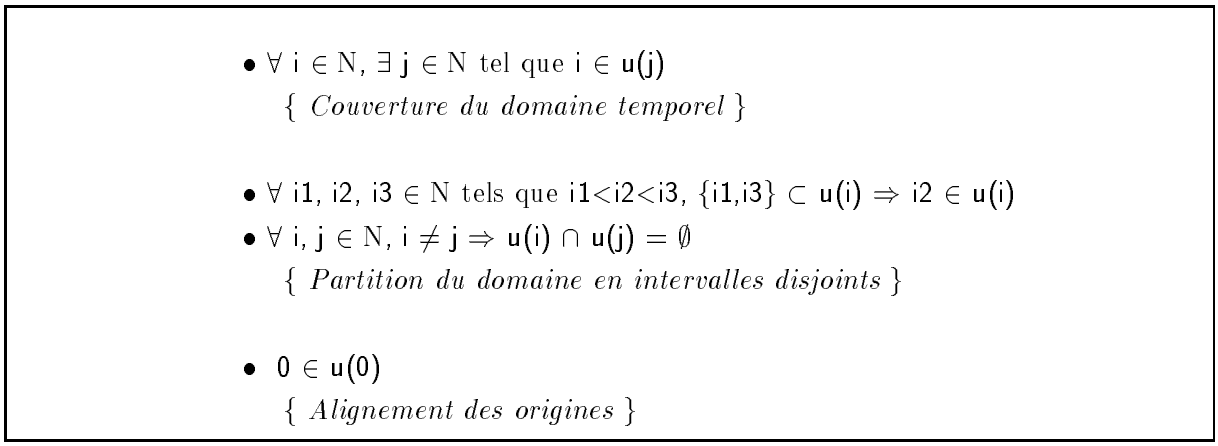


FIG. 2.2 – *Propriétés des unités (d'après [WJS95])*

Chaque partition possible de la droite des réels définit ainsi une *unité d'observation du temps* qui fixe le degré d'approximation employé pour repérer les instants dans le temps c'est-à-dire l'interprétation d'un intervalle de temps comme un instant ponctuel (Cf. figure 2.1). Les unités les plus courantes et en particulier celles correspondant aux partitions usuelles des calendriers sont nommées : millénaire siècle année mois jour semaine heure seconde etc. Comme dans le langage courant le nom d'une unité sert aussi à désigner les grains qu'elle définit.

Une unité d'observation est *régulière* si les intervalles de la partition qui la définit ont tous la même taille comme les heures les jours ou les semaines. Une unité d'observation est *irrégulière* si ses intervalles ont des tailles variables comme les mois ou les années.

La précision la plus fine d'observation du temps admise par un SGBD est définie par une unité régulière appelée *chronon* [JCE+94]. Les autres unités sont alors définies comme une partition de l'ensemble des chronons c'est-à-dire de l'ensemble des entiers naturels plutôt que des réels.

2.1.2 Relation d'ordre *est plus fine* sur les unités d'observations

Lorsque l'on peut faire correspondre de manière biunivoque à tout grain d'une unité u1 un ensemble de grains consécutifs d'une unité u2 u2 est dite *plus fine* que u1 (u1 est *plus grossière* que u2). Par exemple l'unité mois est plus fine que l'unité année (toute année est composée de douze mois consécutifs) et inversement l'unité mois est plus grossière

que l'unité jour (tout mois est composé d'un ensemble de jours consécutifsΓcomportant de 28 à 31 jours selon le cas).

On parle d'unités comparables lorsqu'une unité réalise une partition d'une autre unitéΓ comme c'est le cas entre les mois et les joursΓet d'unités non-comparables lorsque ce n'est pas le casΓpar exemple entre les mois et les semaines (Cf. figure 2.3).

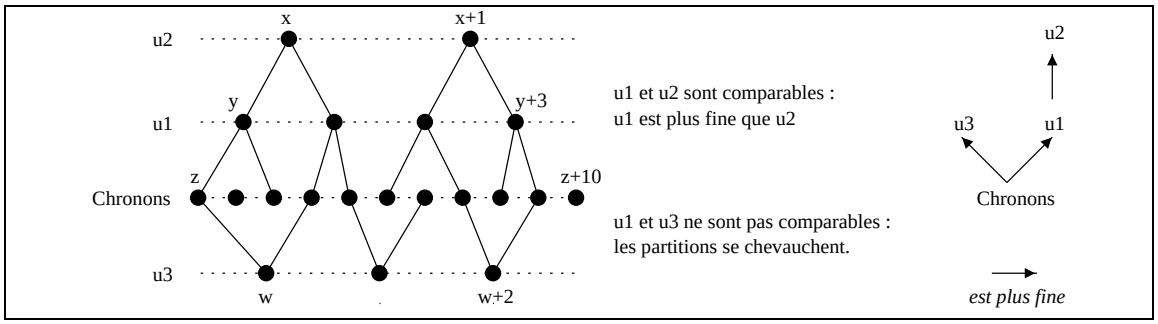


FIG. 2.3 – *Unités comparables et non comparables selon la relation* $\prec$

La relation *est plus fine*Γ notée $\prec$ est une relation d'ordre partiel (Cf. figure 2.4) : par exempleΓles unités mois et semaine définissant des partitions qui se chevauchentΓces unités ne sont pas comparables selon cette relation.

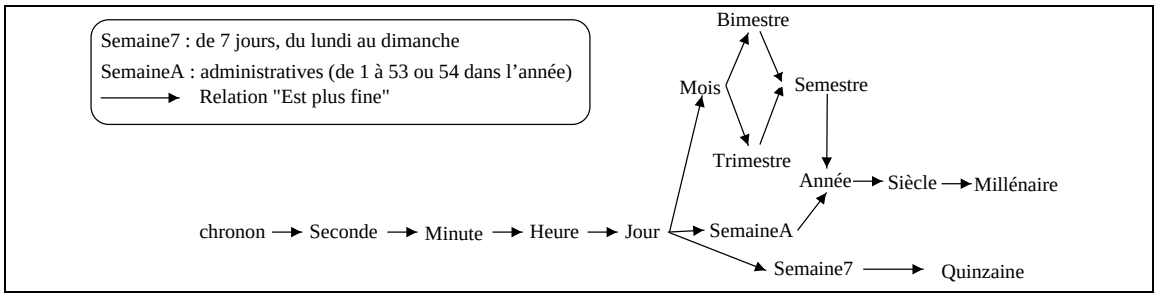


FIG. 2.4 – *La relation est plus fine entre unités d'observation*

Deux unités u et v ont toujours une borne inférieure selon la relation $\prec$Γnotée Ubinf(u,v). Le chronon est l'unité la plus fine de l'ensemble des unités.

2.1.3 Construction de la relation *est plus fine*

A chaque couple d'unités comparables sont associées des fonctions permettant de décrire les conversions correspondantes. Le passage d'une unité à une unité plus fine augmente la précision de l'observation du temps : nous parlons de l'*expansion* d'une unité dans une autre. InversementΓdans le cas du passage d'une unité à une unité plus grossièreΓnous parlons de l'*approximation* d'une unité (Cf. figure 2.5).

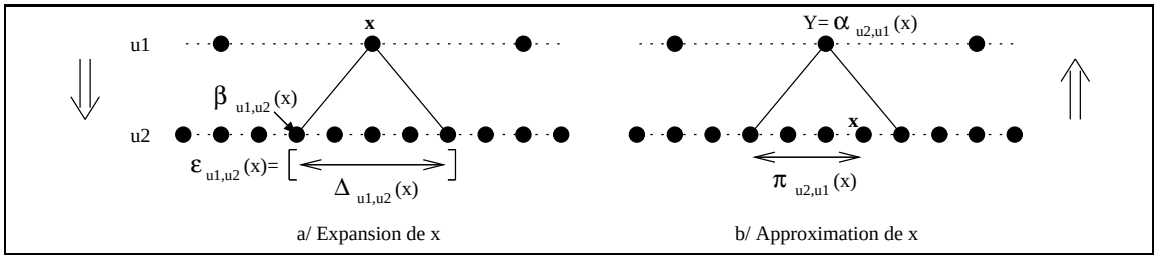


FIG. 2.5 – *Expansion et approximation d'un grain*

Les fonctions liées à l'expansion d'une unité dans une autre sont (u1 et u2 sont deux unités telles que $\mathsf{u2} \prec \mathsf{u1}$) :

- Expansion : $\varepsilon_{\mathsf{u1,u2}}$ dénote la fonction d'*expansion* de u1 dans u2 : $\varepsilon_{\mathsf{u1,u2}}(\mathsf{x})$ est l'intervalle d'entiers formé des numéros de grains de u2 que contient le grain de u1 de numéro x. Par exemple, les douze numéros de mois que comporte une année de numéro donné.

  $\varepsilon_{\mathsf{année,mois}}(1) = [12 \ldots 23]$,
  $\varepsilon_{\mathsf{heure,minute}}(3) = [180 \ldots 239]$

- Dimensionnement : $\Delta_{\mathsf{u1,u2}}$ dénote la fonction de *dimensionnement* de l'expansion de u1 dans u2 : $\Delta_{\mathsf{u1,u2}}(\mathsf{x})$ est la dimension de l'intervalle $\varepsilon_{\mathsf{u1,u2}}(\mathsf{x})$. Par exemple, le nombre de jours d'une année est 366 ou 365, le nombre d'heures dans un jour est 24.

  $\Delta_{\mathsf{année,jour}}(\mathsf{x}) = 365$ ou 366 selon la valeur de x,
  $\Delta_{\mathsf{jour,heure}}(\mathsf{x}) = 24$, quel que soit x

- Bornage : $\beta_{\mathsf{u1,u2}}$ dénote la fonction de *bornage* de l'expansion de u1 dans u2 : $\beta_{\mathsf{u1,u2}}(\mathsf{x})$ est la borne inférieure de l'intervalle $\varepsilon_{\mathsf{u1,u2}}(\mathsf{x})$; $\beta_{\mathsf{u1,u2}}(\mathsf{x}+1) - 1$ en est la borne supérieure.

  $\varepsilon_{\mathsf{u1,u2}}(\mathsf{x}) = [\beta_{\mathsf{u1,u2}}(\mathsf{x}) \ldots \beta_{\mathsf{u1,u2}}(\mathsf{x}+1) - 1]$
  $\beta_{\mathsf{u1,u2}}(0) = 0$;
  $\forall\ \mathsf{x} \geq 0,\ \beta_{\mathsf{u1,u2}}(\mathsf{x}+1) = \beta_{\mathsf{u1,u2}}(\mathsf{x}) + \Delta_{\mathsf{u1,u2}}(\mathsf{x})$

Les fonctions liées à l'approximation d'une unité dans une autre sont (u1 et u2 sont deux unités telles que $\mathsf{u2} \prec \mathsf{u1}$) :

- Approximation : $\alpha_{\mathsf{u2,u1}}$ dénote la fonction d'*approximation* de u2 dans u1 : $\alpha_{\mathsf{u2,u1}}(\mathsf{x})$ est le numéro du grain de u1 dont l'expansion dans u2 contient x. Autrement dit, $\alpha_{\mathsf{u2,u1}}(\mathsf{x})$ est l'entier unique Y tel que $\beta_{\mathsf{u1,u2}}(\mathsf{Y}) \leq \mathsf{x} < \beta_{\mathsf{u1,u2}}(\mathsf{Y}+1)$.

  $\mathsf{Y} = \alpha_{\mathsf{u2,u1}}(\mathsf{x})$ – $\mathsf{x} \in \varepsilon_{\mathsf{u1,u2}}(\mathsf{Y})$
  – $\exists\ \mathsf{Z}$ tel que $\mathsf{x} = \beta_{\mathsf{u1,u2}}(\mathsf{Y}) + \mathsf{Z}$ et $0 \leq \mathsf{Z} < \Delta_{\mathsf{u1,u2}}(\mathsf{Y})$

– Positionnement : $\pi_{u2,u1}$ dénote la fonction de *positionnement* relative dans u2 par rapport à u1 : $\pi_{u2,u1}(x)$ est la position relative Z de x dans l'expansion de son approximation Y. En quelque sorteΓelle représente l'information perdue du fait de la diminution de la précision.

$$\pi_{u2,u1}(x) = x - \beta_{u1,u2}\ (\alpha_{u2,u1}(x)\ )$$

2.1.4 Relation de régularité entre deux unités d'observations

Nous étendons la notion de régularité définie précédemment par rapport au chronon (Cf. section 2.1.1) pour caractériser une propriété particulière des couples d'unités comparables selon la relation $\prec$: u1 et u2 étant deux unités telles que u1 $\prec$ u2Γnous disons que *u1 est régulière par rapport à u2* si tout grain de u1 comporte un nombre fixe de grains de u2.

Nous spécifions un prédicat pour cette relation :

EstRég? : deux unités → un booléen
{ *EstRég?(u, v) – v* $\prec$ *u et* $\Delta_{u,v}$ *est constante* }

Une unité régulière est ainsi un unité régulière par rapport au chrononΓle chronon étantΓpar définitionΓrégulier. Par exempleΓl'année est régulière par rapport au moisΓbien que le mois ne soit pas régulier (par rapport au chronon) ; par contreΓl'année n'est pas régulière par rapport au jourΓbien que le jour soit une unité régulière (Cf. figure 2.6).

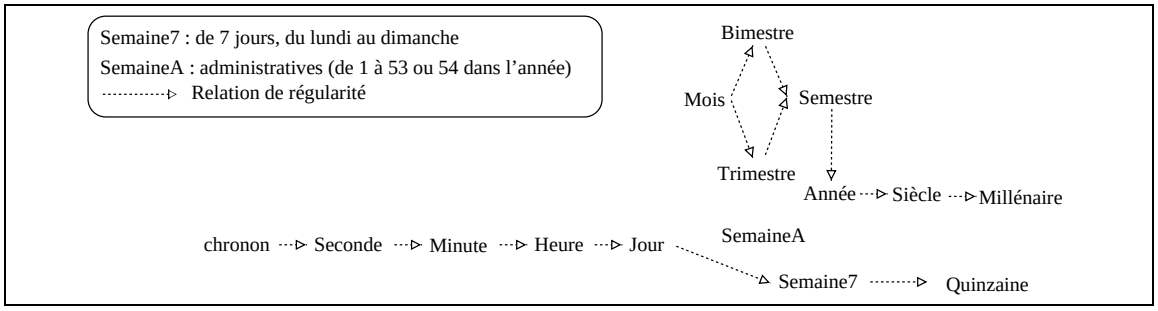


FIG. 2.6 – *La relation de régularité entre unités d'observation*

Si u1 et u2 sont deux unités régulières entre ellesΓ$\Delta_{u1,u2}(x)$ est constant quel que soit le grain xΓon le nomme alors *facteur de grossissement* de u1 dans u2Γnoté $\Gamma_{u1,u2}$. Par exemple :

$$\Delta_{année,mois}(x) = 12 = \Gamma_{année,mois}$$
$$\Delta_{jour,seconde}(x) = 86400 = \Gamma_{jour,seconde}$$

La relation de régularité permet de simplifier les fonctions de conversions de la section précédente (Cf. figure 2.7).

Si EstRég?(u1, u2) alors :

$\beta_{u1,u2}(x) = x * \Gamma_{u1,u2}$

$\varepsilon_{u1,u2}(x) = [x * \Gamma_{u1,u2} \ldots (x+1) * \Gamma_{u1,u2} - 1]$

Par exemple : $\beta_{jour,seconde}(x) = x * 86400$

$\alpha_{u2,u1}(x) = x$ quotient $\Gamma_{u1,u2}$

$\pi_{u2,u1}(x) = x$ reste $\Gamma_{u1,u2}$

FIG. 2.7 – *Simplification des fonctions de conversion pour les couples d'unités réguliers*

2.2 Spécification des types temporels de base

Nous étudions ici les fonctions permettant de manipuler durées et instants. Nous ne développons pas les questions liées aux entrées/sortiesΓelles sont l'objet de la section 2.5.

Tout instant est caractérisé par la durée le séparant de l'origine. Les durées sont réparties en deux types : durées signéesΓdurées non signées.

La figure 2.8 montre la hiérarchie des types selon la relation *EstUn*Γen faisant apparaître le niveau auquel les principales méthodes sont définies : construction (#)Γunité d'observation ($\mathcal{UO}$) et mesure ($\mathcal{M}$) pour les durées et construction (@)Γunité d'observation ($\mathcal{UO}$)Γgrain ($\mathcal{G}$) et durée ($\mathcal{D}$) pour les instants.

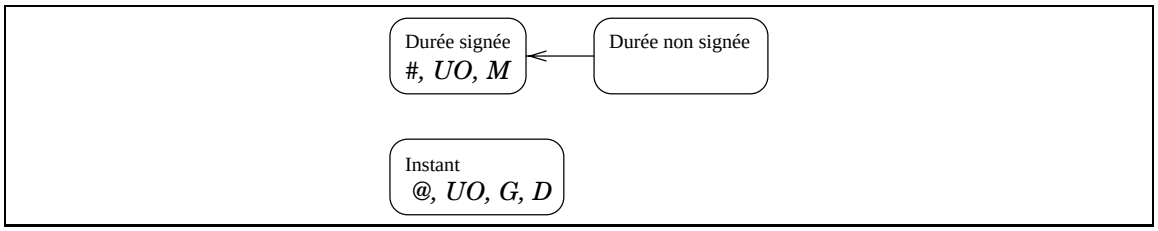


FIG. 2.8 – *Hiérarchie des types temporels de base*

2.2.1 Durées et instants

Une *durée* caractérise l'association d'un entier et d'une unité qui en donne la signification.

Pour permettre l'expression de déplacements dans le temps par rapport à un instant donnéΓvers le passé ou vers le futurΓnous considérons aussi des *durées signées*Γcaractérisées par un entier relatif et une unité d'observation.

Pour faire abstraction des représentations internes ou externesΓil est défini un constructeur (#) et deux sélecteurs associésΓdonnant l'unité d'observation ($\mathcal{UO}$) et la mesure ($\mathcal{M}$) de la durée (Cf. figure 2.9).

La construction d'une durée se fait donc pas l'association d'un entier et d'une unité d'observationΓcomme par exemple 24#jour.

$\#$: un entier , une unité $\longrightarrow$ une durée signée
{ *x # u est une durée observée dans l'unité u et de mesure x* }

(a) Constructeur

$\mathcal{UO}$: une durée signée $\longrightarrow$ une unité
{ *$\mathcal{UO}$(d) est l'unité d'observation de la durée d* }
$\mathcal{M}$: une durée signée $\longrightarrow$ un entier
{ *$\mathcal{M}$(d) est la mesure de la durée d* }

(b) Sélecteurs

$\mathcal{UO}(\mathsf{x} \# \mathsf{u}) = \mathsf{u}$; $\mathcal{M}(\mathsf{x} \# \mathsf{u}) = \mathsf{x}$; $\mathcal{M}(\mathsf{d}) \# \mathcal{UO}(\mathsf{d}) = \mathsf{d}$.

(c) Propriétés

FIG. 2.9 – *Spécification abstraite du type durée*

@ : une durée non signée $\longrightarrow$ un instant
{ *@ (x#u) est l'instant observé dans l'unité u de numéro x dans cette unité.* }

(a) Constructeur

$\mathcal{UO}$: un instant $\longrightarrow$une unité
{ *$\mathcal{UO}$(i) est l'unité d'observation de l'instant i* }
$\mathcal{G}$: un instant $\longrightarrow$ un entier ≥ 0
{ *$\mathcal{G}$(i) est le numéro du grain approximant l'instant i* }
$\mathcal{D}$: un instant $\longrightarrow$ une durée non signée
{ *$\mathcal{D}$(i) est la durée séparant l'instant i de l'origine* }

(b) Sélecteurs

$\mathcal{UO}(@(\mathsf{x}\#\mathsf{u})) = \mathsf{u}$; $\mathcal{G}\ (@(\mathsf{x}\#\mathsf{u}))= \mathsf{x}$; $\mathcal{D}(@(\mathsf{x}\#\mathsf{u})) = \mathsf{x}\#\mathsf{u}$.

(c) Propriétés

FIG. 2.10 – *Spécification abstraite du type instant*

Une durée définie dans une unité régulière peut être interprétée comme la mesure d'un espace de temps. Par contre, si l'unité est irrégulière, comme par exemple 3#année, il n'est pas possible d'interpréter la durée comme la mesure exacte d'un espace de temps. Il est toutefois possible d'en donner une interprétation exacte dans l'unité mois, à savoir 36#mois, l'unité année étant régulière par rapport à l'unité mois.

Dans notre modèle, un *instant* est une approximation d'un espace de temps réel définie par un grain d'une unité d'observation. Un instant est défini par la durée le séparant de l'origine.

Le constructeur d'instant est noté @ (Cf. figure 2.10). Il construit un instant à partir de la durée (exprimée dans son unité d'observation) le séparant de l'origine du temps. Par exemple, @24596#jour dénote l'instant de numéro 24596 dans l'unité jour, c'est-à-dire séparé de l'origine par la durée 24596#jour. De même, @1996#année dénote l'instant de numéro 1996 (c'est-à-dire l'année 1997 dans le calendrier grégorien).

A ce constructeur sont associés trois sélecteurs donnant l'unité d'observation de l'instant ($\mathcal{UO}$), le grain approximant cet instant ($\mathcal{G}$) et la durée non signée le séparant de l'origine ($\mathcal{D}$). La figure 2.10 donne la spécification de ces fonctions.

2.2.2 Fonctions sur les instants et les durées

Le modèle fournit une variété de fonctions classiques traitant d'instants et de durées définis dans la même unité d'observation : prédicats décrivant la relation d'ordre chronologique ($<, \leq, =, \neq, \geq, >$), opérations usuelles sur les durées ($\mathsf{d1} + \mathsf{d2}$, $\mathsf{d1} - \mathsf{d2}$), durée séparant deux instants ($\mathsf{i1} - \mathsf{i2}$), somme algébrique d'une durée et d'un instant ($\mathsf{i} + \mathsf{d}$), etc. (Cf. figure 2.11).

Lorsque c'est possible, ces fonctions sont étendues aux entiers : comparaison d'un instant ou d'une durée avec un entier, ajout ou retrait d'un entier à une durée, etc. (Cf. figure 2.12).

Θ représente un symbole de comparaison sur les entiers ($<\Gamma \leq\Gamma =\Gamma \neq\Gamma \geq\Gamma >$) ; d1 et d2 dénotent deux durées signées de même unité et I1 et I2 deux instants de même unité :

$$d1\ \Theta\ d2 - \ \mathcal{M}(d1)\ \Theta\ \mathcal{M}(d2)\,;\ I1\ \Theta\ I2 - \ \mathcal{D}(I1)\ \Theta\mathcal{D}(I2)$$

Les fonctions déterminant le maximum ou le minimum d'un ensemble de durées ou d'instants sont notées respectivement D_MaxΓD_MinΓet I_Max ΓI_Min.

(a) Egalité, relations d'ordre

d1 et d2 dénotent deux durées signées de même unité et Θ représente un symbole d'addition ($+$) ou de soustraction ($-$).

$$d1\ \Theta\ d2 = (\mathcal{M}(d1)\ \Theta\mathcal{M}(d2))\ \#\ \mathcal{UO}\ (d1)$$

Propriétés : $\mathcal{UO}\ (d1\ \Theta\ d2) = \mathcal{UO}(d1) = \mathcal{UO}\ (d2)\,;\ \mathcal{M}\ (d1\ \Theta\ d2) = \mathcal{M}(d1)\ \Theta\ \mathcal{M}\ (d2)$.

L'addition et la soustraction ont les propriétés algébriques habituelles.

(b) Addition et soustraction de durées

La durée séparant deux instants I1 et I2 de même unité est une durée signée. Nous dénotons cette opération par le symbole de soustraction.

$$I1 - I2 = (\ \mathcal{D}(I1) - \mathcal{D}(I2)\ )$$

Propriétés : $\mathcal{UO}(I1 - I2) = \mathcal{UO}(I1) = \mathcal{UO}(I2)\,;\ \mathcal{M}(I1 - I2) = \mathcal{G}(I1) - \mathcal{G}(I2)$.

(c) Durée séparant deux instants

Un *instant absolu* est un instant définiΓcomme précédemmentΓpar une durée le séparant de l'origine. Par oppositionΓun *instant relatif* est défini par la durée le séparant d'un instant de référence [JCE+94]. Nous dénotons les constructeurs associés par l'un des symboles $+$ ou $-$. Nous les définissons ici pour un instant I et une durée signée d de même unité. Θ représente l'un des symboles $+$ ou $-$:

$$I\ \Theta\ d = @(\ \mathcal{D}(I)\ \Theta\ d\ ) \qquad \{\ \textit{précondition : } \mathcal{D}(I)\ \Theta\ d \geq 0\ \}$$

Propriétés :

$\mathcal{UO}\ (I\ \Theta\ d) = \mathcal{UO}(I) = \mathcal{UO}\ (d)\,;\ \mathcal{D}\ (I\ \Theta\ d) = \mathcal{D}\ (I)\ \Theta\ d$

$I + d + d' = (I + d) + d' = I + (d + d')\,;\ I - d - d' = (I - d) - d'$

(d) Instant relatif

FIG. 2.11 – *Fonctions sur les instants et les durées de même unité d'observation*

Θ représente un symbole de comparaison sur les entiers ($<\Gamma \leq \Gamma = \Gamma \neq \Gamma \geq \Gamma >$) ; $\mathsf{d}\Gamma\mathsf{r}\Gamma\mathsf{I}$ et n dénotent respectivement une durée signéeΓun entier relatifΓun instant et un entier naturel :

- $\mathsf{d}\ \Theta\ \mathsf{r} - \mathsf{d}\ \Theta\ (\mathsf{r} \# \mathcal{UO}(\mathsf{d})) - \mathcal{M}(\mathsf{d})\ \Theta\ \mathsf{r}$;
  $\mathsf{r}\ \Theta\ \mathsf{d} - (\mathsf{r} \# \mathcal{UO}(\mathsf{d}))\ \Theta\ \mathsf{d} - \mathsf{r}\ \Theta\ \mathcal{M}(\mathsf{d})$
- $\mathsf{I}\ \Theta\ \mathsf{n} - \mathcal{D}(\mathsf{I})\ \Theta\ \mathsf{n} - \mathcal{G}(\mathsf{I})\ \Theta\ \mathsf{n}$;
  $\mathsf{n}\ \Theta\ \mathsf{I} - \mathsf{n}\ \Theta\ \mathcal{D}(\mathsf{I}) - \mathsf{n}\ \Theta\ \mathcal{G}(\mathsf{I})$

(a) Egalité, relations d'ordre

ds et r dénotent respectivement une durée signée et un entier relatif ; d et n dénotent respectivement une durée non signée et un entier naturel.

- addition : $\mathsf{ds} + \mathsf{r} = \mathsf{r} + \mathsf{ds} = (\mathcal{M}(\mathsf{ds}) + \mathsf{r}) \# \mathcal{UO}(\mathsf{ds})$
- soustraction : $\mathsf{ds} - \mathsf{r} = (\mathcal{M}(\mathsf{ds}) - \mathsf{r}) \# \mathcal{UO}(\mathsf{ds})$;
  $\mathsf{r} - \mathsf{ds} = (\mathsf{r} - \mathcal{M}(\mathsf{ds})) \# \mathcal{UO}(\mathsf{ds})$
- multiplication : $\mathsf{ds} * \mathsf{r} = \mathsf{r} * \mathsf{ds} = (\mathcal{M}(\mathsf{ds}) * \mathsf{r}) \# \mathcal{UO}(\mathsf{ds})$
- division entière :

$$\mathsf{d\ div\ n} = (\mathcal{M}(\mathsf{d})\ \mathsf{div}\ \mathsf{n}) \# \mathcal{UO}(\mathsf{d}) \qquad \{\ \textit{précondition : } n \neq 0\ \}$$
$$\mathsf{d\ reste\ n} = (\mathcal{M}(\mathsf{d})\ \mathsf{reste}\ \mathsf{n}) \# \mathcal{UO}(\mathsf{d}) \qquad \{\ \textit{précondition : } n \neq 0\ \}$$

(b) Fonctions sur les durées et les entiers

FIG. 2.12 – *Extension aux entiers des fonctions sur les instants et les durées*

Les fonctions de conversions (expansion et approximation) définies sur les couples d'unités (Cf. section 2.1.3) sont étendues aux instants et aux durées. Nous utilisons le nom d'une unité pour dénoter l'expansion ou l'approximation d'un instant (Cf. figure 2.13).

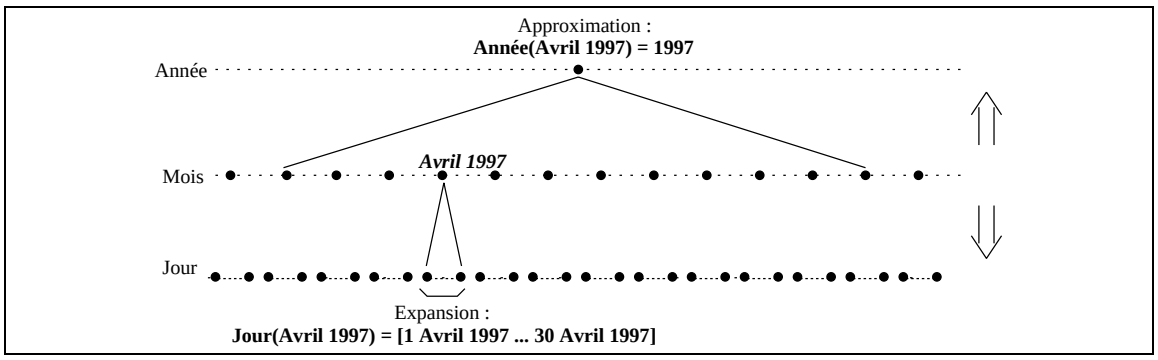


FIG. 2.13 – *Exemple d'expansion et d'approximation d'un instant*

Dans le cas des instants, une expansion fait passer d'un grain d'une unité source à un intervalle de grains de l'unité cible plus fine. Par exemple, l'expansion de l'instant @1#mois (vu comme le deuxième mois depuis l'origine dans le calendrier grégorien) dans l'unité jour résulte en l'intervalle [@31#jour..@59#jour]. Inversement, l'approximation dans l'unité année de l'instant correspondant au 24 avril 1997 (observé à l'unité jour) résulte en l'instant @1996#année (correspondant à l'année 1997).

La figure 2.14 donne la spécification des fonctions d'expansion et d'approximation étendues aux instants.

Dans le cas des durées, la conversion d'une durée d'une unité vers une autre n'est possible que si l'unité source est régulière par rapport à l'unité cible. Par exemple, une durée formulée en années peut être convertie en mois. Par contre, sa conversion en jours n'est pas définie.

2.2.3 Traitement d'instants et de durées d'unités différentes

Nous examinons ici le comportement des opérations définies sur les types de base, et mettant en jeu des valeurs exprimées dans des unités différentes.

Nous illustrons ceci sur le cas des instants et des durées :

- **Traitement de deux instants d'unités différentes**

  I et J dénotent deux instants d'unités respectives u et v ; EI et EJ dénotent les expansions de I et J dans l'unité w borne inférieure de u et v ($\mathsf{EI} = \mathsf{w}(\mathsf{I})$, $\mathsf{EJ} = \mathsf{w}(\mathsf{J})$).

  L'égalité n'est pas définie. Lorsque u et v sont comparables, les deux instants peuvent être situés l'un par rapport à l'autre par le biais des relations entre instants et intervalles. Par exemple si $\mathsf{u} \prec \mathsf{v}$ ($\mathsf{w} = \mathsf{u}$), I peut précéder, suivre ou appartenir à l'intervalle EJ. Ainsi, le 31 janvier 1995 appartient à 1995, précède mars 1996 et suit 1994.

I étant un instant d'unité u1 et u2 une unité plus fine que u1 :

$\beta_{u1,u2}(I)$ est l'instant $@(\beta_{u1,u2}(\mathcal{G}(I)) \# u2)$
$\varepsilon_{u1,u2}(I)$ est l'intervalle d'instants $[\beta_{u1,u2}(I) \ldots \beta_{u1,u2}(I+1) - 1]$
$\Delta_{u1,u2}(I)$ est l'entier $\Delta_{u1,u2}(\mathcal{G}(I))$

I étant un instant d'unité u2 et u1 une unité plus grossière que u2 :

$\alpha_{u2,u1}(I)$ est l'instant $@(\alpha_{u2,u1}(\mathcal{G}(I)) \# u1)$
$\pi_{u2,u1}(I)$ est l'entier $\pi_{u2,u1}(\mathcal{G}(I))$

(a) Fonctions d'expansion et d'approximation étendues aux instants

X étant un intervalle, X^- et X^+ dénotent respectivement ses bornes inférieure et supérieure.
I étant un instant d'unité u, et v étant une unité comparable à u :

- $v \prec u$, $v(I) = \varepsilon_{u,v}(I)$, $v(I)^- = \beta_{u,v}(I)$, $v(I)^+ = \beta_{u,v}(I+1) - 1$
- $v \succ u$, $v(I) = [\alpha_{u,v}(I) \ldots \alpha_{u,v}(I)]$
- $v = u$, $v(I) = [I \ldots I]$

(b) Opérations de conversions

FIG. 2.14 – *Expansion et approximation d'un instant*

Lorsque u et v ne sont pas comparables, les deux instants I et J peuvent être situés l'un par rapport à l'autre par le biais des relations entre intervalles (Cf. section 2.4.1) appliqués aux expansions EI et EJ.

– **Traitement de deux durées d'unités différentes**

Les relations et les opérations (ajout, soustraction) sur des durées d'unités différentes ne sont définies que dans le cas où les deux unités sont régulières l'une par rapport à l'autre. C'est le seul cas où les conversions de durées sont définies (le nombre de jours correspondant à une durée de "2 mois" n'est pas fixe, alors que "2 heures" font toujours "120 minutes"). De cette façon, on est ramené au cas de durées exprimées dans une même unité.

– **Somme d'un instant et d'une durée d'unités différentes**

Nous spécifions ici l'addition et la soustraction entre un instant I d'unité u et une durée non signée d d'unité v. Le résultat est un instant de l'unité w borne inférieure des unités u et v. La somme algébrique d'un instant et d'une durée signée en est déduite, en tenant compte du signe de la durée.

- **Cas 1, $u \succ v$** : exemple "janvier 1995 + 32 jours"

  "janvier 1995 + 32 jours" est défini comme étant le $32^{\text{ème}}$ jour après le mois de janvier 1995Γc'est-à-dire le $32^{\text{ème}}$ jour après le 31 janvier 1995Γsoit le 4 mars 1995.

  $I + d$ est ainsi l'instant obtenu par addition de d à la borne supérieure de l'expansion de I dans l'unité v. De même $I - d$ est déduit de la borne inférieure de cette expansion.

  (1) $u \succ v \Rightarrow I + d = v(I)^+ + d = \beta_{u,v}(I+1) - 1 + d$
  (1') $u \succ v \Rightarrow I - d = v(I)^- - d = \beta_{u,v}(I) - d$

- **Cas 2, $u \prec v$** : exemple "30 janvier 1996 + 3 mois"

  Lorsque l'unité v est régulière par rapport à l'unité uΓla durée peut être convertie dans l'unité de l'instant. Par exempleΓjanvier 1996 + 3 ans = janvier 1996 + 36 mois = janvier 1999.

  (2) $\mathsf{EstRèg}(v, u) \Rightarrow I + d = I + \Gamma_{v,u*}\, d$
  (2') $\mathsf{EstRèg}(v, u) \Rightarrow I - d = I - \Gamma_{v,u*}\, d$

  Dans les autres casΓet du fait de l'irrégularité des unitésΓtoute règle de calcul est arbitraire. Ces cas sont ainsi exclus du modèle.

- **Cas 3, unités non comparables** : exemple "janvier 1995 + 6 semaines"

  L'opération peut être ramenée aux cas précédentsΓen raisonnant sur la borne supérieure de l'expansion de I dans l'unité w borne inférieure de u et v. AinsiΓ"janvier 1995 + 6 semaines" = "31 janvier 1995 + 6 semaines" = "31 janvier 1995 + 42 jours" = "14 mars 1995". La conversion de la durée ne peut être faite que si son unité v est régulière par rapport à l'unité w.

  (3) u et v non comparablesΓ$w = \mathsf{Ubinf}(u, v) \wedge \mathsf{EstRég}(v, w) \Rightarrow I + d = w(I)^+ + d$
  (3') u et v non comparablesΓ$w = \mathsf{Ubinf}(u, v) \wedge \mathsf{EstRég}(v, w) \Rightarrow I - d = w(I)^- - d$

2.3 Représentations multigranulaires d'instants

Les instants peuvent être représentés (de manière interne ou externe) sous *forme multigranulaire*Γen associant une séquence d'entiers et un *système d'unités d'observation.*

Un tel système est une séquence d'unités comparables deux à deuxΓen ordre décroissant selon la relation $\prec$Γcomme par exempleΓles systèmes AMJ = [annéeΓmoisΓjour]Γou JHM = [jourΓheureΓminute]. Un système est dit *régulier* si les unités qui le composent sont régulières deux à deuxΓcomme par exemple le système JHM.

Une valeur temporelle peut être exprimée sous forme multigranulaire dans un système donné dès lors qu'elle est observée dans l'unité la plus fine de ce système. Ceci généralise l'usage courant pour dénoter des dates dans un calendrierΓcomme par exemple 31/8/1995 (système AMJ).

2.3.1 Forme irréductible et formes multigranulaires

La *forme irréductible* d'un grain d'une unité u est le numéro x de ce grain dans l'unité u. Nous parlons du grain x de l'unité u. Les grains étant numérotés à partir de 0Γx représente la durée séparant le grain considéré de l'origine. Par exempleΓla 12032$^{\text{ème}}$ seconde depuis l'origine est le grain 12031 de l'unité seconde.

Une *forme multigranulaire* du grain x de l'unité u dans un système SU $[u_1, ..., u_n]$ (tel que $u_n = u$)Γest une séquence d'entiers naturels $[x_1, ..., x_n]$Γchacun d'eux caractérisant une durée dans chacun des u_i. Le grain x est obtenu en calculant les instants relatifs successifs à partir de l'origine et en ajoutant les entiers x_1 à x_n : $x = (((0 + x_1) + x_2) + ...) + x_n$. Ces calculs successifs portent sur des valeurs d'unités différentesΓune conversion est réalisée à chaque foisΓcomme cela est défini dans la section 2.2.3.

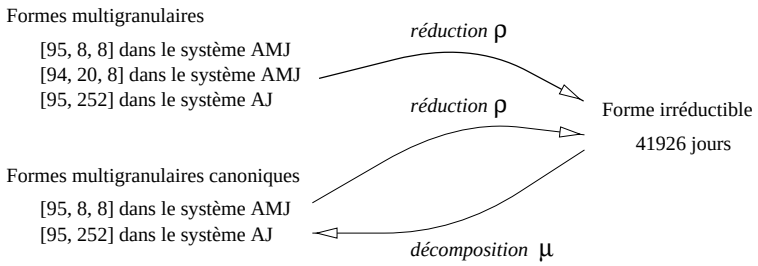
[0Γ2Γ79Γ91] est une forme multigranulaireΓdans le système JHMSΓdu grain 12031 de l'unité seconde. Le détail du calcul est le suivant :

$0+0 = 0$ { *dans l'unité jour* }
$\quad$ Heure(0) = 0
$\quad$ $0+2 = 2$ { *dans l'unité heure* }
$\quad\quad$ Minute(2) = 120
$\quad\quad$ $120 + 79 = 199$ { *dans l'unité minute* }
$\quad\quad\quad$ Seconde(199) = 11940
$\quad\quad\quad$ $11940 + 91 = 12031$ { *dans l'unité seconde* }

La *forme multigranulaire canonique* du grain x de l'unité u dans un système SU (dont l'unité d'observation est u)Γest la séquence d'entiers naturels telle que chaque entier est la position relative du grain par rapport à son approximation dans l'unité immédiatement plus grossière dans SU. Par exempleΓ[0Γ3Γ20Γ31] est la forme multigranulaire canonique dans le système JHMS du grain 12031 de l'unité seconde : pour atteindre ce grainΓil s'est écoulé 3 heures 20 minutes et 31 secondes depuis l'origine et c'est donc la 32$^{\text{ème}}$ seconde de la 21$^{\text{ème}}$ minute de la 4$^{\text{ème}}$ heure du 1$^{\text{er}}$ jour.

2.3.2 Conversion entre formes

La relation existant entre formes irréductible et multigranulaires d'un grain est définie par deux opérations de conversion (Cf. figure 2.15) : une *décomposition* convertit une forme irréductible en sa forme multigranulaire canonique dans un système d'unités ; une *réduction* convertit toute forme multigranulaire en sa forme irréductible. La composition de ces opérations permet des conversions plus générales d'un système d'unités dans un autre. On peut ainsi convertir "le 9 septembre 1996" en "le jour 253 de l'année 1996" ou en "le lundi de la semaine 37 de l'année 1996"Γen passant par leur forme irréductible commune (41926 joursΓen supposant que l'origine est le 1/1/1900).



(a) Conversions entre formes (en considérant l'origine au 1/1/1900)

• μ_{SU} dénote la fonction de *décomposition* dans le système SU : $\mu_{SU}(x)$ est la forme multigranulaire canonique dans SU du grain x de l'unité la plus fine de SU.

• ρ_{SU} dénote la fonction de *réduction* du système SU : $\rho_{SU}(X)$ est la forme irréductible du grain de forme multigranulaire X dans SU.

• Propriétés (le constructeur de séquence [ ] et l'opération & de concaténation sont définis en annexe C) :

(1) $\mu_{[u]}\ (x) = [x]$

(2) $\mu_{[u_1,\ ...,\ u_n]}\ (x) = \mu_{[u_1,\ ...,\ u_{n-1}]}\ (\alpha_{u_n,\ u_{n-1}}\ (x))\ \&\ [\pi_{u_n,\ u_{n-1}}\ (x)]$ { $n \geq 2$ }

ou, si $[u_1, ..., u_n]$ est régulier :

(2') $\mu_{[u_1,\ ...,\ u_n]}\ (x) = [\alpha_{u_n,\ u_1}\ (x)]\ \&\ \mu_{[u_2,\ ...,\ u_n]}\ (\ \pi_{u_n,\ u_1}\ (x))$ { $n \geq 2$ }

$= [x \text{ quotient } \Gamma_{u_n,\ u_1}]\ \&\ \mu_{[u_2,\ ...,\ u_n]}\ (\ x \text{ reste } \Gamma_{u_n,\ u_1})$

(3) $\rho_{[u]}([x]) = x$

(4) $\rho_{[u_1,\ ...,\ u_n]}\ ([x_1, ..., x_n]) = \beta_{u_{n-1},\ u_n}\ (\rho_{[u_1,\ ...,\ u_{n-1}]}\ ([x_1, ..., x_{n-1}])) + x_n$ { $n \geq 2$ }

ou bien :

(4') $\rho_{[u_1,\ ...,\ u_n]}\ ([x_1, ..., x_n]) = \rho_{[u_2,\ ...,\ u_n]}\ ([\beta_{u_1,\ u_2}(x_1) + x_2, \ ...\ , x_n])$ { $n \geq 2$ }

ou, si $[u_1, ..., u_n]$ est régulier :

(4") $\rho_{[u_1,\ ...,\ u_n]}\ ([x_1, ..., x_n]) = \beta_{u_1,\ u_n}(x_1) + \rho_{[u_2,\ ...,\ u_n]}\ ([x_2, ..., x_n])$ { $n \geq 2$ }

$= x_1 * \Gamma_{u_1,\ u_n} + \rho_{[u_2,\ ...,\ u_n]}\ ([x_2, ..., x_n])$

(4"') $\rho_{[u_1,\ ...,\ u_n]}\ ([x_1, ..., x_n]) = \Gamma_{u_{n-1},\ u_n} * (\rho_{[u_1,\ ...,\ u_{n-1}]}\ ([x_1, ..., x_{n-1}])) + x_n$

(b) Fonctions de décomposition et de réduction

FIG. 2.15 – *Formes irréductibles et formes multigranulaires*

2.4 Ensembles d'instants

Parmi les diverses représentations d'un ensemble d'instants, nous distinguons les *intervalles d'instants*, les *D_séquences (séquences d'intervalles discontinus)* et les *séquences d'instants périodiques*.

Nous présentons ici ces différents types ainsi que les fonctions qui leur sont associées.

2.4.1 Intervalles

Un *intervalle d'instants*, caractérisé par ses deux bornes, est la séquence en ordre chronologique de tous les instants observables dans l'unité d'observation, entre ces deux instants.

Il est possible de construire un intervalle avec deux instants (constructeur [I...J]) ou avec un instant et une durée (constructeur [I | d]). Les sélecteurs associés permettent d'obtenir ses bornes, sa durée et son unité d'observation (Cf. figure 2.16).

Le modèle fournit une variété de fonctions traitant des intervalles : un ensemble de prédicats décrivant des relations sur les intervalles (à la manière de Allen [All83], Cf. section 1.1.3) et également des relations entre instants et intervalles (Cf. figure 2.17). Il y a enfin des opérateurs (Cf. figure 2.18) : les opérateurs ensemblistes et des opérateurs de translation et de redimensionnement des intervalles.

Comme pour les instants, nous étendons les fonctions d'expansion et d'approximation aux intervalles.

L'expansion dans une unité u2 d'un intervalle observé dans une unité u1, u2 $\prec$ u1, est l'intervalle de u2 déduit de l'expansion dans u2 des bornes de l'intervalle. De même, l'approximation dans une unité u1 d'un intervalle observé dans une unité u2, u2 $\prec$ u1, est un intervalle de u1 déduit de l'approximation dans u1 des bornes de l'intervalle. La définition des fonctions est la suivante :

$$\varepsilon_{u1,u2}([I...J]) = [\beta_{u1,u2}(I) \ldots \beta_{u1,u2}(J+1) - 1]$$
$$\alpha_{u2,u1}([I...J]) = [\alpha_{u2,u1}(I) \ldots \alpha_{u2,u1}(J)]$$

Nous utilisons le nom d'une unité pour dénoter l'opération d'expansion ou d'appoximation d'un intervalle (X est un intervalle d'unité u et v est une unité comparable à u) :

Si $v \prec u$, $v(X) = \varepsilon_{u,v}(X)$;
Si $v \succ u$, $v(X) = \alpha_{u,v}(X)$;
Si $v = u$, $v(X) = X$.
Dans les trois cas : $(v(X))^- = (v(X^-))^-$, $(v(X))^+ = (v(X^+))^+$.

Par exemple, jour([1914 ... 1918]) = [1/1/1914 ... 31/12/1918], et année([juin 1997 ... septembre 1997]) = [1997 ... 1997].

I et J dénotant deux instants de même unité d'observationΓsi I $\leq$ JΓ[I...J] dénote l'intervalle fermé de borne inférieure I et de borne supérieure J et sinon un intervalle vide ; [ ] dénote l'intervalle vide. De plusΓI et d dénotant un instant et une durée non signée de même unitéΓ [I | d] dénote l'intervalle [I...I+d−1].

(a) Constructeurs

– L'intervalle vide est caractérisé par le prédicat EstVide? :

EstVide? : un intervalle $\longrightarrow$ un booléen $\quad$ { *vrai* – *l'intervalle donné est vide* }

– Les bornes d'un intervalle.
X dénotant un intervalle non videΓX^- dénote sa borne inférieure. Si X est borné à droiteΓ X^+ dénote sa borne supérieure :

$[I...J]^- = I$, $[I...J]^+ = J$, $[X^- ... X^+] = X$

– La dimension d'un intervalle borné est le nombre d'instants qu'il comporte.

Dim : un intervalle borné $\longrightarrow$ un entier ≥ 0
{ $Dim(X) = si\ EstVide?(X)\ alors\ 0\ sinon\ \mathcal{G}(X^+) - \mathcal{G}(X^-) + 1$ }

– L'unité d'observation d'un intervalle non vide est l'unité d'observation de ses bornes.

$\mathcal{UO}$: un intervalle non vide $\longrightarrow$ une unité
{ $\mathcal{UO}(X) = \mathcal{UO}(X^-) = \mathcal{UO}(X^+)$ }

– La durée d'un intervalle est mesurée dans l'unité de l'intervalle. Sa mesure est la dimension de l'intervalle :

$\mathcal{D}$: un intervalle borné $\longrightarrow$ une durée non signée
{ $\mathcal{D}(X) = Dim(X) \# \mathcal{UO}(X)$ }

(b) Sélecteurs

FIG. 2.16 – *Spécification abstraite du type intervalle*

Nous adoptons les treize relations définies par Allen [All83] (Cf. section 1.1.3) :

relation X r Y	définition	réciproque Y r^{-1} X
X *précède* Y (X<Y)	$X^+ < Y^-$	Y *suit* X (X>Y)
X *rejoint* Y	$X^+ = Y^-$	Y *est rejoint par* X
X *chevauche* Y	$X^- < Y^- \wedge Y^- < X^+ \wedge X^+ < Y^+$	Y *est chevauché par* X
X *est dans* Y	$X^- > Y^- \wedge X^+ < Y^+$	Y *englobe* X
X *débute* Y	$X^- = Y^- \wedge X^+ < Y^+$	Y *est débuté par* X
X *termine* Y	$X^- > Y^- \wedge X^+ = Y^+$	Y *est terminé par* X
X *égale* Y (X=Y)	$X^- = Y^- \wedge X^+ = Y^+$	Y *égale* X

Par ailleurs, nous définissons aussi les prédicats suivants :

X *est contigu à* Y : Contigu?(X, Y) – $X^+ = Y^- - 1$ { *relation asymétrique* }

X et Y *sont disjoints* : Disjoints?(X,Y) – $X < Y \vee X > Y$ { *relation symétrique* }

X *est inclu dans* Y (relation d'ordre) au sens large, notée $\subseteq$, ou strict, notée $\subset$.

La réciproque (au sens large ou strict) est Y *contient* X (notée $\supseteq$ ou $\supset$) :

$X \subseteq Y$ – $X^- \geq Y^- \wedge X^+ \leq Y^+$

{ – *X est dans Y ou X débute Y ou X termine Y ou X égale Y* }

$X \subset Y$ – $(X^- > Y^- \wedge X^+ \leq Y^+) \vee (X^- \geq Y^- \wedge X^+ < Y^+)$

{ – *X est dans Y ou X débute Y ou X termine Y* }

(a) Relations sur les intervalles

I et X dénotent un instant et un intervalle de même unité.

I *précède* X, noté $I < Y$ – $I < X^-$

I *suit* X, noté $I > Y$ – $I > X^+$

I *appartient à* X, noté $I \in Y$ – $X^- \leq I \wedge I \leq X^+$

(b) Relations entre intervalles et instants

FIG. 2.17 – *Prédicats sur les intervalles*

X et Y dénotent deux intervalles non vides observés selon la même unité.

– Intersection notée $X \cap Y$:

$$X \cap Y = \text{si Disjoints?}(X, Y) \text{ alors } [\,] \text{ sinon } [\text{I\_Max}(X^-, Y^-) \ldots \text{I\_Min}(X^+, Y^+)]$$

– Couverture (plus petit intervalle contenant chacun des deux intervalles) :

$$\text{Couverture}(X, Y) : [\text{I\_Min}(X^-, Y^-) \ldots \text{I\_Max}(X^+, Y^+)]$$

Si les deux intervalles ne sont pas disjoints leur couverture est leur union ensembliste.

(a) Opérations internes sur les intervalles

Une *translation* déplace un intervalle dans le temps dans le passé ou dans le futur d'une durée donnée. Un *redimensionnement* allonge ou raccourcit la dimension de l'intervalle d'une durée donnée en déplaçant l'une des deux bornes.
X dénote un intervalle non vide et d une durée signée de même unité.

– Translations : on surcharge les symboles + et −.

$X+\,d$ dénote l'intervalle $[X^- + d \ldots X^+ + d]$ $\quad \{\ \textit{précondition :}\ X^- + d \geq 0\ \}$
$X-\,d$ dénote l'intervalle $[X^- - d \ldots X^+ - d]$ $\quad \{\ \textit{précondition :}\ X^- - d \geq 0\ \}$

– Redimensionnements

- borne inférieure fixe
  $X ++\,d$ dénote l'intervalle $[X^- \ldots X^+ + d]$ $\quad \{\ \textit{précondition :}\ X^+ + d \geq 0\ \}$
  $X +-\,d$ dénote l'intervalle $[X^- \ldots X^+ - d]$ $\quad \{\ \textit{précondition :}\ X^+ - d \geq 0\ \}$
- borne supérieure fixe
  $X --\,d$ dénote l'intervalle $[X^- - d \ldots X^+]$ $\quad \{\ \textit{précondition :}\ X^- - d \geq 0\ \}$
  $X -+\,d$ dénote l'intervalle $[X^- + d \ldots X^+]$ $\quad \{\ \textit{précondition :}\ X^- + d \geq 0\ \}$

(b) Translation et redimensionnement d'un intervalle

FIG. 2.18 – *Opérations sur les intervalles*

2.4.2 D_séquences : séquences d'intervalles discontinus

Une *D_séquence* (*temporal element* dans [JCE+94]Γ*coalesced temporal element* dans [BCTP95]Γ*interval* ou *non convex interval* dans [Lad86]Γ*generalized interval* dans [Yu 83]) est une séquence finie d'intervalles non vides de même unitéΓdeux à deux disjoints et non contigusΓet ordonnés selon la relation précède :

$[X_1, X_2, \ldots, X_n]$ est une D_séquence $-\ \forall\ k \in [1 \ldots n-1],\ X_k^+ < X_{k+1}^- - 1$.

Tout ensemble fini d'instants peut être représenté par une D_séquenceΓet les D_séquences héritent des opérations ensemblistes. Les opérations de constructionΓles sélecteurs associés et les opérations de manipulations des D_séquences sont présentés dans la figure 2.19.

2.4.3 Séquences d'instants périodiques

Une *séquence d'instants périodique* est une séquence d'instants observés selon la même unité et en ordre chronologique tels que deux instants consécutifs soient séparés par la même durée non nulleΓla *période* de la séquence. Les séquences d'instants périodiques généralisent les intervalles (séquences périodiques de période 1). Un constructeurΓ des sélecteursΓdes relations et des opérations de changement de période sont définies sur les séquences périodiques (Cf. figure 2.20).

La *forme compacte* d'un ensemble d'instants E de même unité est la D_séquence DS formée des instants de E.
Dans le cas où les instants de E sont d'unités différentesΓla forme compacte de E est définie dans l'unité borne inférieure de ces unités à partir des expansions des instants donnés.
Nous définissons ainsi les deux opérations de conversion suivantes

LaFC : un ensemble d'instants $\longrightarrow$ une D_séquence
Lext :une D_séquence $\longrightarrow$ un ensemble d'instants
{ *Propriétés :* Lext(LaFC(E)) = E ; LaFC(Lext(DS)) = DS }

(a) Forme compacte d'un ensemble d'instants

- La borne inférieure (resp. supérieure) d'une D_séquence DSΓ notée DS^- (resp. DS^+) est la borne inférieure du premier intervalle (resp. la borne sup. du dernier) : $DS^- = (\text{premier}(DS))^-$; $DS^+ = (\text{dernier}(DS))^+$
- La dimension d'une D_séquence est le nombre d'instants de son extension :
  Dim : une D_séquence $\longrightarrow$ un entier ≥ 0 $\{Dim(DS) = cardinal(Lext(DS))\}$
- L'unité d'observation d'une D_séquence est l'unité commune à tous ses intervalles (extension de la fonction $\mathcal{UO}$ aux D_séquences).
- La durée (cumulée des intervalles) d'une D_séquence est mesurée dans l'unité d'observation de la D_séquence. Sa mesure est la dimension de la D_séquence :
  $\mathcal{D}$: une D_séquence $\longrightarrow$ une durée non signée $\{\mathcal{D}(DS) = Dim(DS)\#\mathcal{UO}(DS)\}$

(b) Sélecteurs

- Ajout et suppression d'un instant à une D_séquence. DS et I dénotent respectivement une D_séquence et un instant de même unité. Nous surchargeons les symboles + et − dans ce contexte :
  DS + I = LaFC(Lext(DS) ∪ {I}) et DS − I = LaFC(Lext(DS)\ {I}).
- Opérations ensemblistes. Les D_séquences héritent de toutes les opérations ensemblistes. DS1 et DS2 dénotant deux D_séquences et Θ représentant une opération ensembliste $(\cup\Gamma\cap\Gamma\setminus)$: DS1 Θ DS2 = LaFC(Lext(DS1) Θ Lext(DS2)). Ces opérations sont également étendues aux intervalles : DS Θ X = LaFC(Lext(DS) Θ Lext(X)).
- Itérateurs sur les D_séquences. Les D_séquences héritent de toutes les opérations sur les séquences (Cf. annexe C) et en particulier d'itérateurs permettant d'appliquer une opération à tous les intervalles d'une D_séquence. Nous en montrons quelques exemples dans la section 2.4.5.

(c) Autres opérations

FIG. 2.19 – *Opérations sur les D_séquences*

[I | d, n] dénote la séquence périodique d'instant initial IΓde période d et de dimension nΓ où I et d dénotent un instant et une durée non signée de même unité et n dénote un entier naturel :

[I | d, 0] dénote une séquence vide.
[I | d,n+1] = {J | ∃ k ∈ [0...n] tel que J = I+k∗d} { $n > 0$ }

Ce constructeur est défini par les relations de récurrence suivantes :

(1) [I | d, 0] = [ ]
(2) [I | d, n+1] = [I | d,n] & [I + n∗d] { $n \geq 0$ }

De plusΓ[I | d, ...[dénote une séquence périodique infinie non bornée à droite. Le prédicat EstBornée? caractérise une séquence périodique bornée.

(a) Constructeur

- Extension aux séquences périodiques des sélecteurs portant sur les intervalles :
  - Borne inférieure et borne supérieure : $[I \mid d, n]^- = I$; $[I \mid d, n]^+ = I + (n-1)* d$
  - Dimension : Dim [I | d, n] = n
  - Unité d'observation : $\mathcal{UO}$ ([I | dΓn]) = $\mathcal{UO}$ (I)
  - Durée : $\mathcal{D}$ ([I | dΓn]) = (n−1)∗d + 1
- La période d'une séquence périodique est une durée :
  Pér : une séquence périodique ⟶ une durée > 0 {*Pér([I | d,n]) = d*}
- Propriété (SP dénotant une séquence périodique) :
  EstBornée?(SP) ∧ ¬EstVide?(SP) ⇒ [SP⁻ | Pér(SP), Dim(SP)] = SP

(b) Sélecteurs

Nous étendons certaines des relations définies sur les intervalles. S1 et S2 dénotant deux séquences périodiques bornées non vides de même unité :

S1 *chevauche* S2 – $S1^- < S2^- \wedge S2^- < S1^+ \wedge S1^+ < S2^+$
S1 = S2 – Lext(S1) = Lext(S2)
S1 < S2 – $S1^+ < S2^-$
S1 ⊆ S2 – $S1^- \geq S2^- \wedge S1^+ \leq S2^+$

(c) Relations

Le changement de périodeΓnoté ↔ permet la *compression* (d < 0Γdiminution de la période) ou l'*étalement* (d > 0Γaugmentation de la période) d'une séquence périodique. [I | dΓn] ↔ d' = [I | d+ d'Γn] {*précondition : d+d' ≥ 0*}

(d) Changement de période

FIG. 2.20 – *Opérations sur les séquences d'instants périodiques*

2.4.4 Structuration d'ensembles d'instants, Calendriers

Nous proposons ici les moyens de structurer les ensembles d'instants afin de pouvoir représenter des calendriers [LMF86ΓLEW96].

Nous définissons ainsi une *n_séquence* de T (T étant un type quelconque)Γnotée $[T]^n$Γ comme suit :

1_séquence de T ≡ séquence de T : $[T]^1 \equiv [T]$
(n+1)_séquence de T ≡ séquence de n_séquences de T : $[T]^{n+1} \equiv [\ [T]^n\ ]$

Les n_séquences peuvent être représentées par des arbres dont les feuilles sont des 1_séquences de T et les noeuds non-terminaux sont des constructeurs de séquences (Cf. figure 2.21).

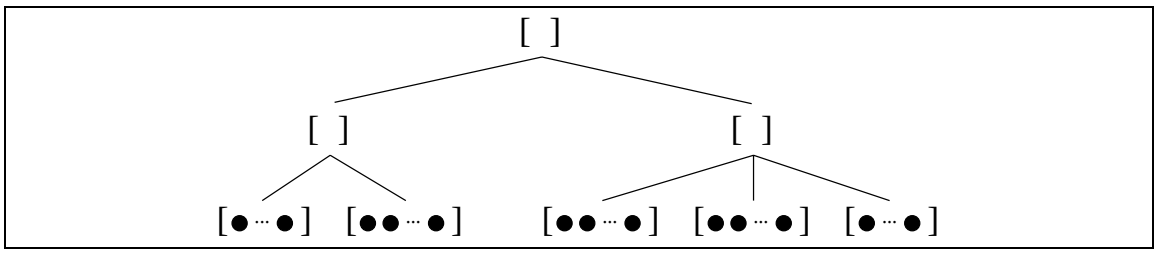


FIG. 2.21 – *Représentation arborescente d'une 3_séquence*

Nous définissons des fonctions sur les séquences d'instants pour les structurer et obtenir des calendriers (n_séquences). La partition selon un système d'unitésΓnotée $\mathcal{P}_{SU}$Γ regroupe successivement les instants ayant même approximation dans les unités du système d'unités. L'expansion structurée selon un système d'unitésΓnotée $\mathcal{E}_{SU}$Γeffectue les expansions successives d'une séquence d'instants dans les unités du système d'unités. Leurs spécifications sont données dans la figure 2.23 en terme des fonctions ApplicationΓ DebEtFin et LesGroupes.

La fonction Application applique une fonction élément par élémentΓà tous les éléments d'une séquence. La fonction DebEtFin découpe une séquence en deux selon une propriété (Cf. annexe C).

La fonction LesGroupes construit une partition d'une séquence S selon une fonction f (chacune des séquences de la séquence résultat regroupe les éléments consécutifs de S ayant la même image par f) (Cf. figure 2.22).

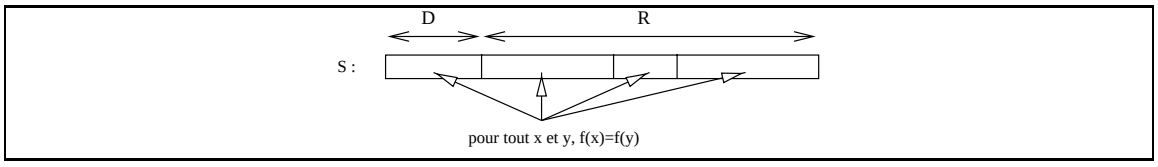


FIG. 2.22 – *LesGroupes : partition d'une séquence selon une fonction*

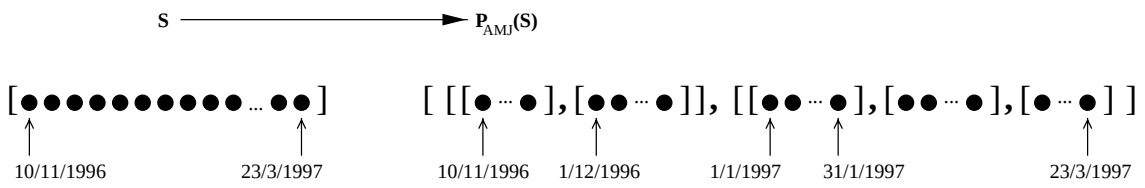
La spécification de la fonction LesGroupes est la suivante :

LesGroupes : [T], (T $\longrightarrow$ T') $\longrightarrow$ [[T]]

{ *LesGroupes (S, f) est la séquence de séquences composées des éléments de S telles que tous les éléments d'une sous-séquence ont même image par f.* }

(1) LesGroupes ([],f) = []

(2) LesGroupes ($e \circ S$, f) = soit $<D, R> =$ DebEtFin (S, $\lambda x \bullet f(x) \neq f(e)$)
dans $(e \circ D) \circ$LesGroupes (R, f)



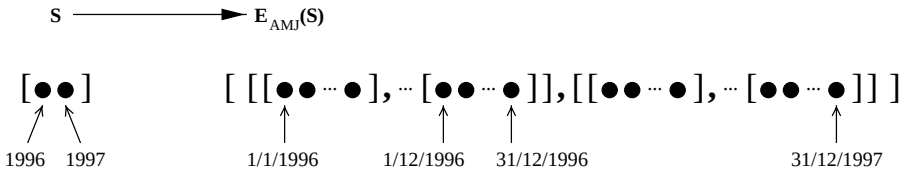
$\mathcal{P}_{SU}$: [instant(v)] $\longrightarrow$ [instant(u_n)]^n

{ *SU est un système d'unités* $[u_1, ..., u_n]$, *tel que* $u_n \succ v$ }

$\mathcal{P}_{[u]}$(S) = LesGroupes (S, u) { *précondition :* $u > v$ }

$\mathcal{P}_{[u_1, ..., u_n]}$(S) = Application ($\mathcal{P}_{[u_1]}$(S), $\mathcal{P}_{[u_2, ..., u_n]}$) { *précondition :* $u_n > v$ }

(a) Partition



$\mathcal{E}_{SU}$: [instant(v)] $\longrightarrow$ [instant(u_n)]^n

{ *SU est un système d'unités* $[u_1, ..., u_n]$, *tel que* $v > u_1$ }

$\mathcal{E}_{[u]}$(S) = Application (S, u) { *précondition :* $v \succ u$ }

$\mathcal{E}_{[u_1, ..., u_n]}$(S) = Application ($\mathcal{E}_{[u_1]}$(S), $\mathcal{E}_{[u_2, ..., u_n]}$) { *précondition :* $v > u_1$ }

(b) Expansion

FIG. 2.23 – *Construction de calendriers*

2.4.5 Exemples de manipulation de séquences d'instants

Les types précédents sont des sous-types du type séquence (Cf. figure 2.24).

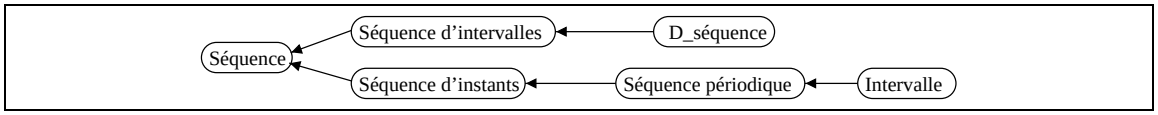


FIG. 2.24 – *Hiérarchie de types associés aux séquences d'instants*

Nous montrons ici divers exemples d'expressions d'instants et de séquences d'instants. Nous utilisons pour cela des fonctions générales sur les séquences. La fonction Application applique une fonctionΓélément par élément à tous les éléments d'une séquence. La fonction Sélection extrait d'une séquence tous les éléments vérifiant une propriété donnée. La fonction Appli_Sel est la composition des fonctions Application et Sélection (application d'une fonction aux éléments résultant d'une sélection). La fonction LaFin fournit la fin d'une séquence à partir du premier élément vérifiant une propriété donnée. Toutes ces fonctions sont définies dans l'annexe C.

Les instants sont formulés en considérant que l'origine est l'année 1 du calendrier grégorien.

– Le numéro de jour de semaine (1 à 7) d'un instant I (unité jour) :

$$\pi_{\text{jour,semaine7}}(\text{I}) + 1$$

– Le numéro de mois (de 1 à 12) d'un instant I (unité jour) :

$$\pi_{\text{mois,an}}(\text{mois}(\text{I})^{-}) + 1$$

– Les années pleines entre deux instants I et J (unité jour)ΓI et J inclus :

$$[\text{année}(\text{I})^{-} + (\text{si jour}(\text{année}(\text{I})^{-})^{-} = \text{I alors 0 sinon 1})$$
$$\ldots \text{ année}(\text{J}) \; {}^{-}{-} \; (\text{si jour}(\text{année}(\text{J})^{-}) \; {}^{+} = \text{J alors 0 sinon 1})]$$

– Les lundis de 1996 (le lundi est le jour de numéro 0) :

$$\text{Sélection (jour(@1995\#année)}, \lambda \text{j} \bullet \pi_{\text{jour,semaine7}}(\text{j}) = 0)$$

– Les week−end de 1996 :

$$\text{Appli\_Sel(jour(@1995\#année)}, \lambda \text{j} \bullet \pi_{\text{jour,semaine7}}(\text{j}) = 5, \lambda \text{j} \bullet [\text{j, j+1}] \text{)}$$

– Le premier lundi de 1996 (on sait qu'il existe) :

$$\text{premier(LaFin(jour(@1995\#année)}, \lambda \text{j} \bullet \pi_{\text{jour,semaine7}}(\text{j}) = 0))$$

- Le mardi de la première semaine pleine de Novembre 1996 (le jour suivant le premier lundi de Novembre 1996) :

  $1 + \text{premier}(\text{LaFin}(\text{jour}(@[1995, 10]\#[\text{année, mois}]), \lambda j \bullet \pi_{\text{jour,semaine7}}(j) = 0))$

- Les jours de la semaine administrative contenant le 15 mars 1996 sous forme multigranulaire dans le système AMJ :

  $\text{Application } (\text{jour}(\text{semaineA}(@[1995, 2, 14]\#\text{AMJ})^{-}), \mu_{AMJ})$

- Le calendrier de l'année 1996Γdécomposé en mois et en jours :

  - En utilisant les fonctions d'expansion :

    $\text{Application } (\text{mois}(@1995\#\text{année}), \text{jour})$

  - En utilisant les fonctions sur les calendriers :

    $\mathcal{E}_{[\text{année, mois, jour}]}\ ([@1995\#\text{année}])$

- Le calendrier de l'année 1996Γdécomposé en semaines et en jours :

  - En utilisant les fonctions d'expansion :

    $\text{Application } (\text{semaineA}(@1995\#\text{année}), \text{jour})$

  - En utilisant les fonctions sur les calendriers :

    $\mathcal{E}_{[\text{année, semaineA, jour}]}\ ([@1995\#\text{année}])$

- Le calendrier de janvier 1996 décomposé en semaines et en jours :

  - En utilisant les fonctions d'expansion :

    $\text{Appli\_Sel}(\text{semaineA}(@1995\#\text{année}),$
    $\lambda s \bullet \text{jour}(s) \subseteq \text{jour}(@[1995, 0]\#[\text{année, mois}]),$
    $\text{jour})$

    { *Il faut d'abord convertir l'année complète en semaines, et ensuite ne retenir que les semaines du mois de janvier* }

  - En utilisant les fonctions sur les calendriers :

    $\mathcal{E}_{[\text{semaineA, jour}]}\ ([@[1995, 0]\#[\text{année, mois}])$

2.5 Représentation externe des valeurs temporelles

Les formes externes des valeurs temporelles sont multiples. Elles sont liées aux unités utilisées pour exprimer les valeurs temporelles. Le modèle du temps proposé offrant un ensemble extensible d'unités d'observation du temps, il est nécessaire de rendre également extensible le système de formes externes.

Dans cette section, nous présentons un travail relatif au traitement des formes externes des instants [Dum96, CD97]. Il doit être complété pour traiter des formes externes d'ensembles d'instants et également des durées.

2.5.1 Diversité des formes externes

Les modes d'expression des instants sont nombreux et variés comme le témoigne le corpus suivant dans lequel toutes les expressions désignent le même instant :

- "5/8/1996" (dans le format européen)
- "8/5/1996" (dans le format américain)
- "1996-08-05" (selon le standard ISO-8601:1988 [ISO88])
- "5 Août 1996" (en Français)
- "August 5, 1996" (en Anglais)
- "5/8/96" (sous-entendu $20^{\text{ème}}$ siècle)
- "lundi semaine 33 année 1996" (dans le système d'unités [année, semaine, jour])

Cette variété est due à divers facteurs : calendriers utilisés, contraintes typographiques, contexte linguistique et culturel, habitudes personnelles, implicites, etc. Ces facteurs peuvent être classifiés en deux grandes catégories : ceux qui dépendent du calendrier utilisé et ceux qui concernent le contexte linguistique et culturel dans lequel on se place.

Le calendrier détermine l'ensemble d'unités et de systèmes d'unités dans lequels sont exprimées les valeurs temporelles. Le calendrier grégorien, par exemple, fait intervenir les unités *jour*, *mois*, *année* et *siècle*, et les systèmes d'unités [*année*, *mois*, *jour*] (AMJ) et [*année*, *mois*] entre autres.

Le contexte linguistique de son côté détermine la correspondance entre les valeurs temporelles et leurs représentations externes ainsi que la structure lexicale et syntaxique de ces représentations. Dans la section suivante, nous introduisons la notion de *format* pour modéliser cette correspondance.

2.5.2 Formats

Nous regroupons toutes les propriétés des représentations externes des valeurs temporelles dépendant du contexte linguistique et culturel sous la notion de *format*.

Considérons par exemple le *format européen* des dates. Les formes externes de ce format sont les chaînes de caractères de la forme "13/6/1997" ou "16/6/1997". L'interprétation de la forme externe "13/6/1997" de ce format est l'instant de forme multigranulaire [1996Γ5Γ12] dans le système d'unités AMJ. Par ailleursΓles formes externes de ce format dénotent des instants observés à la granularité du *jour*.

Le domaine d'un format est un langage régulier sur l'alphabet constitué des caractères alphanumériquesΓdes signes de ponctuationΓdes noms des unités et des noms des grains des unités. Il est donc entièrement caractérisé par la donnée d'une expression régulière sur cet alphabet. Par exempleΠl'expression régulière décrivant le domaine du format européen des dates est $(\mathbf{1} \mid \mathbf{2} \ldots \mid \mathbf{31})$ / $(\mathbf{1} \mid \mathbf{2} \ldots \mid \mathbf{12})$ / $(\mathbf{0} \mid \mathbf{1} \ldots \mid \mathbf{9})^+$.

A tout format est associé un système d'unités qui permet d'accepter et d'interpréter une forme externe dans ce format. Par exempleΓle système d'unités associé au format européen est le système AMJΓassocié aussi au format américain.

En parcourant une forme externe de gauche à droite on peut construire une liste d'entiers correspondant aux grains d'unités rencontrés. Cette liste contient tous les éléments de la forme multigranulaire de l'interprétation de cette forme externeΓmais ce n'est pas toujours sa forme multigranulaire parce qu'elle n'est pas forcément ordonnée correctement. Par exempleΓen effectuant ce parcours sur la forme externe "13/6/1997" on obtient la liste [12Γ5Γ1996] alors que la forme multigranulaire dans le système AMJ de son interprétation est [1996Γ5Γ12] dans le format européen ou [1996Γ12Γ5] dans le format américain.

Pour résoudre ce problèmeΓnous associons à chaque format une permutation sur $[1, n]$ (où n est le nombre d'unités du système associé à ce format) qui définit la correspondance entre les éléments de la liste de grains d'unités extraite d'une forme externe et la forme multigranulaire de son interprétation. AinsiΓsi l'image de 1 par cette permutation est 3Γ l'élément d'indice 1 dans la liste extraite de la forme externeΓsera l'élément d'indice 3 dans la forme multigranulaire recherchée.

Par exempleΓla liste de grains extraite de la forme externe "13/6/1997" étant [12Γ5Γ 1996]Γet la permutation associée au format européen des dates étant [3Γ2Γ1]Γon déduit que la forme multigranulaire de l'instant dénoté par "13/6/1997" dans le système d'unités AMJ est [1996Γ5Γ12].

Une fois que nous disposons de la forme multigranulaire d'une valeur temporelle dans un système d'unitésΓnous pouvons obtenir cette valeur temporelle grâce aux constructeurs associés aux types temporels (Cf. figure 2.25).

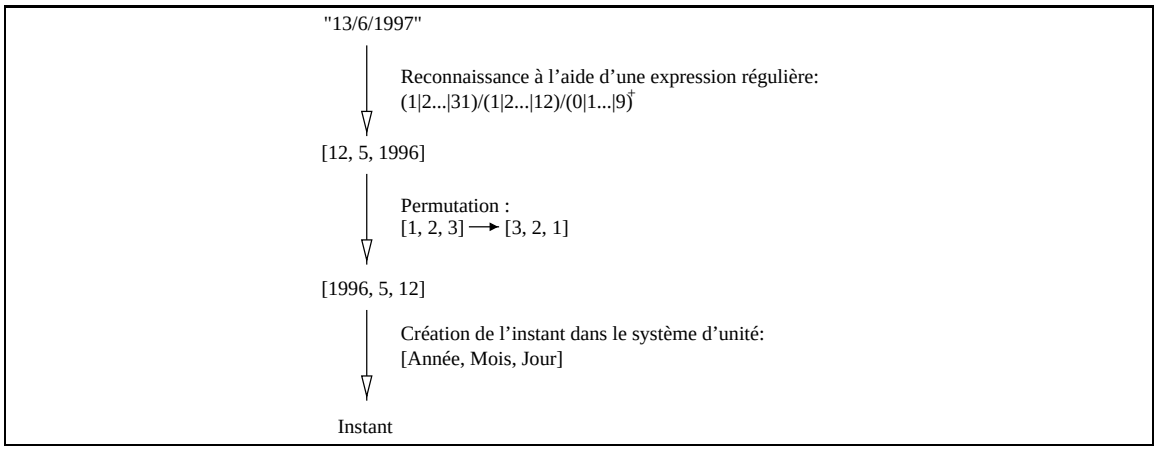


FIG. 2.25 – *Reconnaissance d'une forme externe d'un instant*

Nous sommes donc en mesure de calculer l'interprétation d'une forme externe connaissant le système d'unités et la permutation associée à son format (fonction Rec Cf. figure 2.26). Inversement nous pouvons aussi produire la forme externe d'une valeur temporelle dans un format donné à partir de sa forme multigranulaire dans le système d'unités de ce format (fonction FE Cf. figure 2.26). Ces considérations nous permettent de dire qu'un format est entièrement caractérisé par la donnée d'une expression régulière d'un système d'unités et d'une permutation (Cf. figure 2.26).

Format : < expression régulière, système d'unités, permutation >

Fonctions associées aux formats :

Rec : format, chaîne $\longrightarrow$ instant

{ *Rec(f, c) est l'instant construit à partir de c analysée suivant le format f.* }

FE : format, instant $\longrightarrow$ chaîne

{ *FE(f, i) est la chaîne de caractères correspondant à la forme externe de i dans le format f.* }

FIG. 2.26 – *Format*

Une forme externe peut appartenir au domaine de plusieurs formats à la fois. La forme externe "5/8/1996" peut correspondre à une date exprimée dans le format européen et à une autre date dans le format américain. Pour traiter ce type d'ambiguïté nous regroupons les formats au sein de *systèmes de formats*. Un système de formats est un ensemble de formats de domaines deux-à-deux disjoints. Ainsi au sein d'un système de formats les formes externes ont une interprétation unique.

La forme externe d'instant "5/8/96" présentée dans le corpus ci-dessus fait référence à l'instant de forme multigranulaire [1996 8 5] dans le système d'unités AMJ et non pas

à celui de forme multigranulaire [96Γ8Γ5] qu'on obtiendrait normalement. Ceci est dû à l'usage courant qui est de repérer un instant non pas par rapport à l'origine (imposée par le modèle ou par le système)Γmais par rapport à un autre instant de référence qu'on appelera *origine translatée du temps*. En l'occurrenceΓil s'agit dans cet exemple du 1er janvier 1900.

Un *seuil de changement d'origine* est également défini. Si l'instant obtenu à partir de l'origine translatée est postérieur au seuil de changement d'origineΓalors c'est l'origine standard qui est utilisée. AinsiΓdans l'exempleΓsi la valeur reconnue est "5/8/1996"Γet que l'instant correspondant est calculé à partir du 1er janvier 1900Γl'instant résultant est le 5/8/3896. Il faut donc fixer le seuilΓpar exempleΓau 31/12/1999.

Chapitre 3

Tempos : Historiques

Nous l'avons vu au chapitre 1Γles SGBD temporels associent les historiques au niveau des attributs ou au niveau des n-uplets (ou des objets selon le modèle de données). L'association au niveau des n-uplets permet d'avoir une vision globale d'une entité à un instant donné. Mais dans ce casΓla nature de l'évolution propre à chaque attribut est perdue : ceux-ci peuvent évoluer ou non de manière synchroneΓéventuellement à des granularités différentes.

Au niveau des applicationsΓon peut vouloir observer l'évolution au niveau des attributs ou au niveau des n-uplets. Dans le modèle relationnelΓle respect de la première forme normale impose le choix n-uplet. Dans les modèles à objetsΓla structuration offerte permet les deux options. Nous choisissons comme base du modèle l'étude de l'historisation des attributsΓsachant que l'historisation des objets pourra être traitée à partir de là.

Nous traitons dans ce chapitre des attributs historiques. Nous décrivons les caractéristiques temporelles des attributsΓnous classifions les historiques en fonction des modalités de leur mise à jour et nous étudions leurs représentations. Nous présentons enfin les opérations d'interrogation et de mises à jour des historiques.

3.1 Proposition pour un modèle d'historiques

Le *statut d'évolution* d'un attribut caractérise son comportement dans le temps et la manière avec laquelle il est observé. Il y a trois cas possibles :

- Un attribut d'un objet est *constant* si sa valeur structurelle ne doit jamais changer.
- S'il peut évoluerΓil est alors :
  - *Fugitif* si l'on ne s'intéresse qu'à sa valeur structurelle la plus récenteΓqu'elle traduise une correction ou une évolution.

– *Historique* si l'on veut en observer l'évolution au cours du temps en retenant une suite d'états pertinents.

Un *historique* est la valeur associée à un attribut historique.

3.1.1 Caractéristiques des historiques

Une association temporelle est l'association d'un attribut et d'un couple <M, V> : V est la valeur structurelle observée de l'attribut au moment MΓun moment étant ici un instant ou un ensemble d'instants. Le couple <M, V> est appelé *instantané*.

Au niveau le plus abstraitΓun attribut historique est une fonction à domaine d'entités et à valeur historiqueΓun historique étant lui même une fonction à domaine temporel [WD93].

Un historique est caractérisé par :

– La *dimension temporelle*Γqui fixe la signification des estampilles temporelles par rapport au monde réel (temps de validité) ou par rapport au SGBD (temps de transaction).

– L'*unité d'observation*Γqui fixe la granularité de l'observation de son évolution. C'est l'unité des instants datant les instantanés. Dans le cas d'attributs bitemporelsΓdeux unités sont définiesΓl'une pour le temps de validité et l'autre pour le temps de transaction.

– Le *domaine temporel* (de validité et/ou de transaction)Γnoté DomTΓqui est le domaine de la fonction historique sous-jacente. Il peut être modifié dynamiquement par des opérations spécifiques [1].

– Le type de la valeur structurelle de l'attribut.

3.1.2 Modalités de construction des historiques

Un historique est construit par un ensemble de mises à jour sur la base qui fournissent les données traduisant l'évolution de l'attribut considéré ou des corrections le concernant. Ces données peuvent ne porter que sur une partie du domaine temporelΓinstants choisis selon une périodicité propre à la nature de l'attribut. Dans ce casΓles valeurs associées aux autres instants du domaine sont définies selon diverses modalités que nous précisons ici.

1. Le domaine temporel d'un attribut est un sous-ensemble de la période de vie de l'entité considérée. On peut ainsi décrire des situations où une propriété n'est pas définie temporairement. C'est par exemple le cas si l'on considère les périodes d'emploi d'un salarié d'une entreprise, entrecoupées de périodes d'absence.

Les mises à jour d'un attribut historique construisent un ensemble d'instantanés que nous appelons *effectifs* pour marquer le fait qu'ils sont issus d'une saisie de données. Leurs valeurs sont dites *effectives* et leurs instants forment le *domaine (temporel) effectif* de l'historique noté DomEff. Par opposition le *domaine potentiel* de l'historique noté DomPot est la différence entre le domaine temporel et le domaine effectif[2]. Il correspond aux instants pour lesquels les mises à jour n'ont pas fourni de données. Les valeurs et instantanés correspondant sont dit *potentiels* pour marquer que leur valeur n'est pas pour autant nécessairement absente dans la réalité et peut être déterminée au moment de la consultation de l'historique par une *interpolation temporelle* sur l'ensemble des instantanés effectifs. Dans le cas où la mise à jour doit traduire une absence effective d'informations dans la réalité de l'application la valeur effective est dite *absente*.

Nous distinguons quatre types d'attributs historiques selon la manière de déterminer les valeurs potentielles :

- Attribut historique *discret* : les valeurs potentielles sont interprétées comme des valeurs absentes (exemple : historique des ventes d'un magasin selon une périodicité fixée a priori).

- Attribut historique *régulier* : c'est un cas particulier d'attribut discret ne comportant que des instantanés effectifs. Son domaine temporel par nature dynamique est égal à son domaine effectif (exemple : historique des opérations de crédit sur un compte bancaire).

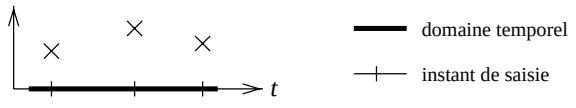
- Attribut historique *en escalier* : l'attribut considéré évolue de telle manière que ses valeurs restent stables entre deux modifications (exemple : historique du solde d'un compte en banque).

- Attribut historique *interpolé* : les valeurs potentielles sont déterminées par une fonction d'interpolation temporelle (exemple : historique d'un processus physique).

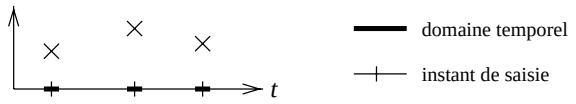
  Un attribut historique observé selon le temps de transaction est par nature en escalier. Un attribut historique bitemporel peut être vu comme un attribut en escalier sur le temps de transaction dont les valeurs structurelles sont des historiques sur le temps de validité.

La figure 3.1 illustre les divers types d'historiques. Pour chaque cas la figure montre le domaine temporel de l'historique et les points effectifs de saisie.

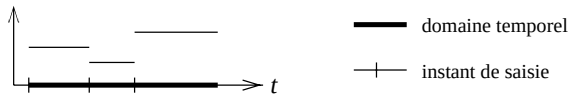
2. Dans [SS93] on distingue de même les *time points* qui forment le domaine temporel et les *data points* qui forment le domaine effectif.



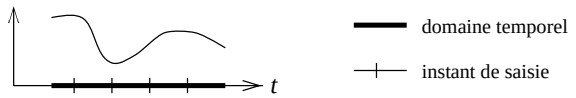
(a) Historique discret



(b) Historique régulier



(c) Historique en escalier



(d) Historique interpolé

FIG. 3.1 – *Les divers types d'attributs historiques*

3.1.3 Représentation des historiques

Toute séquence d'instantanés est appelée une *chronique*. Son domaine temporel est implicitement donné par l'ensemble de ses instantanés. Une chronique peut être interprétée comme un historique régulier et inversement. D'un autre coté les représentations des historiques sont basées sur celles des chroniques.

Toutes les valeurs d'un historique ne sont pas nécessairement stockées. TEMPOS propose deux types de représentations :

- Représentation *en extension* : l'historique est décrit en faisant abstraction des mises à jour qui l'ont défini ; c'est une représentation du graphe de la fonction temporelle sous-jacente sous forme d'une chronique de tous ses instantanés. Elle est déduite des instantanés effectifs de l'historique et de l'interprétation de ses valeurs potentielles.

- Représentation *en compréhension* : l'historique est décrit par son domaine temporel la chronique de ses instantanés effectifs et une fonction d'interpolation temporelle. Dans le cas des historiques discrets ou en escalier cette fonction est fournie par le système. Dans le cas des historiques interpolés elle est donnée explicitement par l'utilisateur.

  Ainsi la valeur d'un historique en compréhension à un instant i donné est définie suivant le cas par :

- $i \in DomT$
  - $i \in DomEff$: Valeur structurelle de l'instantané
  - $i \in DomPot$: Interpolation selon le type d'historique :
    - Discret : Valeur absente
    - Escalier : Valeur de l'instantané précédent
    - Interpolé : Valeur interpolée avec la fonction fournie
- $i \notin DomT$: Non défini

La figure 3.2 illustre l'interprétation d'une chronique de valeurs saisies selon que l'attribut correspondant est défini comme étant discretΓen escalier ou interpolé.

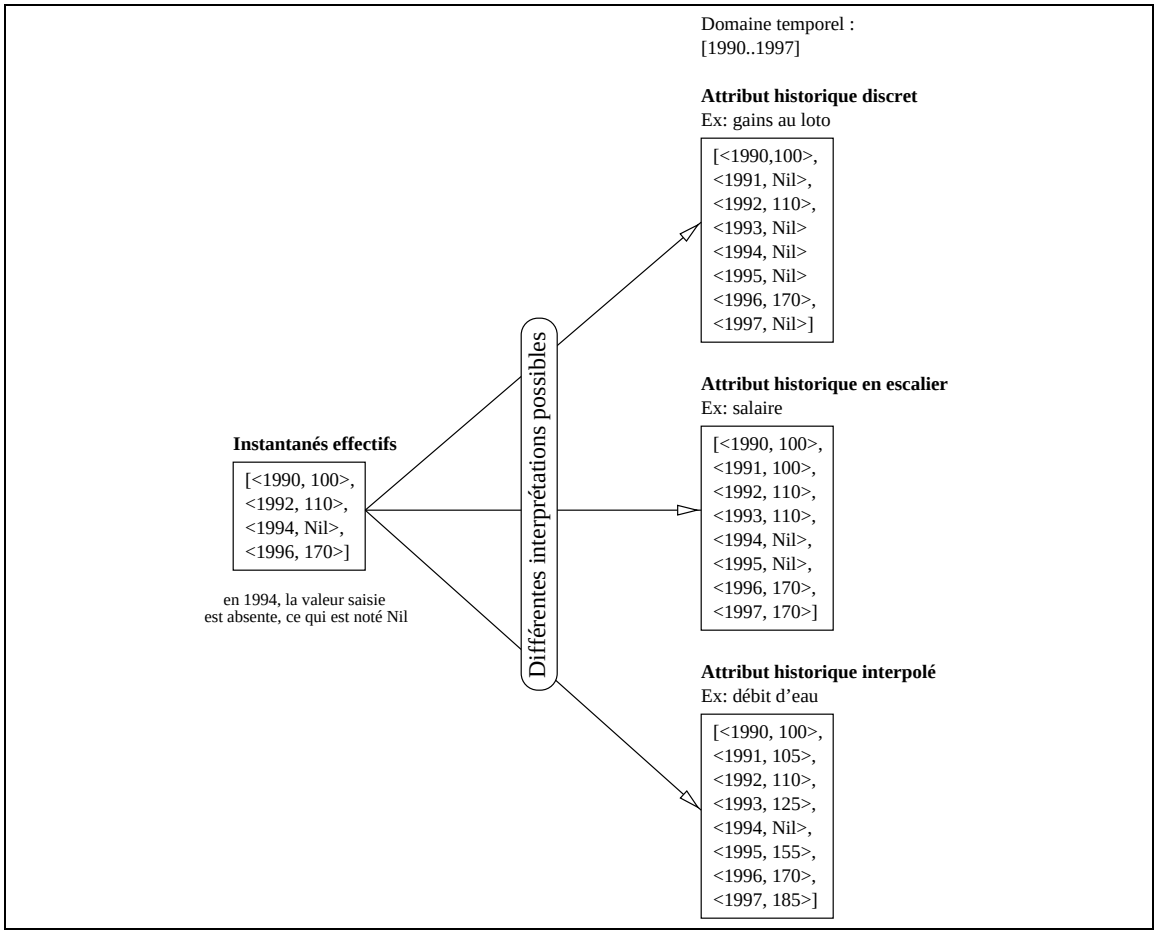


FIG. 3.2 – *Interprétation des instantanés effectifs suivant le type d'attribut historique*

3.1.4 Représentation des chroniques

Pour un instantané <M, V>ΓM peut êtreΓun instant ou un ensemble d'instants. En effetΓune même valeur structurelle peut être datée par un seul instant ou par un ensemble d'instants pour traduire que la valeur observée est la même à chacun des instants de cet ensemble.

Nous parlons de *I_instantanés* de *X_instantanés* ou de *D_instantanés* selon que le moment d'observation est un instant un intervalle ou une D_séquence.

De même nous particularisons trois formes de représentation d'une chronique sous forme d'une séquence d'instantanés (Cf. figure 3.3) :

- Une *I_chronique* est une séquence de I_instantanés en ordre chronologique. Une même valeur structurelle peut y apparaître plusieurs fois.

- Une *X_chronique* est une séquence de X_instantanés en ordre croissant selon la relation *précède*[3] sur les intervalles. Ces intervalles sont disjoints deux à deux et une même valeur structurelle peut apparaître plusieurs fois. Une X_chronique est dite sous forme *canonique* si le regroupement par intervalles est maximal c'est-à-dire si deux X_instantanés consécutifs dont les intervalles sont contigus ont des valeurs structurelles différentes.

- Une *D_chronique* est une séquence de D_instantanés dont les valeurs structurelles sont distinctes deux à deux.

I_chronique :	[<1, v1>, <2, v1>, <3, v1>, <4, v2>,
	<5, v2>, <7, v3>, <8, v1>, <9, v1>],
X_chronique :	[<[1...3], v1>, <[4...5], v2>, <[7...7], v3>, <[8...9], v1>]
D_chronique :	[<[[1...3], [8...9]], v1>, <[[4...5]], v2>, <[[7...7]], v3>]

FIG. 3.3 – *Représentations d'une chronique (les instants sont figurés par des entiers)*

Les modèles étudiés dans le chapitre 1 proposent des représentations équivalentes à l'une ou l'autre de ces trois formes. Dans TEMPOS un même historique quelque soit son type (discret en escalier interpolé) peut être vu au travers de I_chronique X_chronique ou D_chronique.

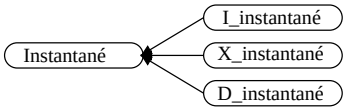
3. [I..J] précède [K..L] $\Longleftrightarrow$ J < K

3.1.5 Types associés au modèle d'historiques

Nous définissons les types temporels nécessaires à la formalisation du modèle d'historiques.

– Le type Instantané(u,T) caractérise les instantanés observés à l'unité u et dont la valeur structurelle est de type T. Il est spécialisé en trois sous-types I_instantané(u,T)Γ X_instantané(u,T) et D_instantané(u,T)Γse distinguant selon que la valeur temporelle est un instantΓun intervalle ou une D_séquence (Cf. figure 3.4a).

  Les informations portées par un instantané (quelle que soit la spécialisation) sont accessibles par les trois fonctions suivantes : S étant un instantanéΓUO (S)ΓVS (S) et VT (S) dénotent respectivement l'unité d'observationΓla valeur structurelle et la valeur temporelle (instantΓintervalle ou D_séquence selon la spécialisation) de S. Les prédicats définis sur les valeurs temporelles sont étendus aux instantanés. Par exempleΓ deux X_instantanés sont dans la relation *précède* si leurs intervalles le sont ; deux D_instantanés sont disjoints si leurs D_séquences le sont ; etc.



(a) Hiérarchie des types

TYPE Moment(u) : Instant(u) ∪ Intervalle(u) ∪ D_séquence(u)
TYPE Instantané(u, T) : <Moment(u), T>
TYPE I_instantané(u, T) : <Instant(u), T>
TYPE X_instantané(u, T) : <Intervalle(u), T>
TYPE D_instantané(u, T) : <D_séquence(u), T>

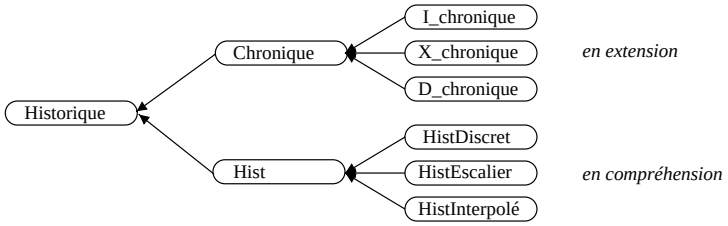
UO : Instantané(u, T) ⟶ unité
{ *unité d'observation* }
VS : Instantané(u, T) ⟶ T
{ *valeur structurelle : VS(<M, V>) = V* }
VT : Instantané(u, T) ⟶ Moment(u)
{ *valeur temporelle : VT(<M, V>) = M* }

(b) Description des types et fonctions

FIG. 3.4 – *Le type Instantané*

– Le type Historique (u, T) caractérise les historiques observés à l'unité u et dont les valeurs structurelles sont de type T. Il est spécialisé en deux sous-types nommés Chronique (u, T) et Hist (u, T) correspondant respectivement aux représentations en extension et en compréhension des historiques (Cf. figure 3.5a).

  Les informations portées par un historique (quelle que soit la représentation) sont accessibles par les fonctions suivantes (Cf. figure 3.5b) : H étant un historiqueΓUO (H) et DomT (H) dénotent respectivement l'unité d'observation et le domaine temporel de H ; DomS (H) est le domaine structurel de HΓensemble des valeurs structurelles présentes dans les instantanés composant H. I étant un instantΓVS (H, I) est la valeur structurelle observée à l'instant I. InversementΓV étant une valeur structurelleΓIO (H, V) est l'ensemble des instants auxquels la valeur structurelle V a été observée.



(a) Hiérarchie des types

UO : Historique (u, T) $\longrightarrow$ unité
{ *unité d'observation* }
DomT : Historique (u, T) $\longrightarrow$ {Instant (u)}
{ *domaine temporel* }
VS : Historique (u, T), Instant (u) $\longrightarrow$ T { *Précondition : I ∈ DomT (H)* }
{ *VS (H, I) est la valeur structurelle à l'instant I, Nil si elle est absente.* }
IO : Historique (u, T), T $\longrightarrow$ {Instant (u)}
{ *instants d'observation* }
{ $IO\ (H,\ V) = \{I \in DomT\ (H) \mid VS\ (H,\ I) = V\}$ }
DomS : Historique (u, T) $\longrightarrow$ {T}
{ *domaine des valeurs structurelles* }
{ $DomS\ (H) = \{V \mid \exists\ I \in DomT\ (H)$ *tel que* $VS\ (H,\ I) = V\}$ }

Relation d'appartenance :
<I, V> étant de type I_instantané (u, T) et H de type Historique (u, T),
<I, V> $\in_i$ H $\Longleftrightarrow \exists$ I $\in$ DomT (H) tel que VS (H, I) = V.

(b) Fonctions sur les historiques

FIG. 3.5 – *Le type Historique*

– Le type Hist est spécialisé en trois sous-typesΓHistDiscretΓHistEscalier et HistInterpoléΓ qui correspondent à la classification en historiques discretsΓen escalier et interpolé (les historiques réguliers sont des chroniques). Ils sont munis de fonctions spécifiques (Cf. figure 3.6) : H étant de type HistΓChronEff (H) est la chronique des instantanés effectifs de H ; DomPot (H) et DomEff (H) dénotent respectivement les domaines potentiel et effectif de H.

ChronEff : Hist (u, T) $\longrightarrow$ Chronique (u, T)
{ *instantanés effectifs* }
DomPot, DomEff : Hist (u, T) $\longrightarrow$ {Instant (u)}
{ *domaines potentiel et effectif* }
{ *DomEff (H) = DomT (ChronEff (H)) ; DomT (H) = DomPot (H) ∪ Deff (H) ; DomPot (H) ∩ Deff (H) = ∅* }

(a) Fonctions sur les Hist

TYPE HistDiscret (u, T) : <{Instant (u)}, Chronique (u, T)>
{ *<D, C> = HD étant de type HistDiscret (u, T) :*
DomT (HD) = D ; ChronEff (HD) = C ;
VS (H, I) = si I ∈ DomEff (HD) alors VS (C, I) sinon Nil }
TYPE HistEscalier (u, T) : <{Instant (u)}, Chronique (u, T)>
{ *<D, C> = HE étant de type HistEscalier (u, T) :*
DomT (HE) = D ; ChronEff (HE) = C ;
en posant IC = I_Chr (C), si I précède l'instant du premier instantané de IC, VS (HE, I) = Nil. Sinon, Z étant l'instantané de IC d'instant égal ou directement antérieur à I, VS (HE, I) = VS (Z) }

TYPE Interpolation (u, T) : Instant (u, T), Chronique (u, T) $\longrightarrow$ T
{ *fonction d'interpolation temporelle* }

TYPE HistInterpolé (u, T) : < {Instant (u)}, Chronique (u, T), Interpolation (u, T)>
{ *<D, C, FI> = HI étant de type HistInterpolé (u, T) :*
DomT (HI) = D ; ChronEff (HI) = C ;
VS (HI, I) = si I ∈ DomEff (HI) alors VS (C, I) sinon FI (I, C) }

(b) Sous-types de Hist

FIG. 3.6 – *Historiques en compréhension : le type Hist*

– Les types I_chroniqueΓX_chronique et D_chronique se distinguent par le type des instantanés composant la chronique (Cf. figure 3.7a). La fonction I_Chr (respectivement X_Chr et D_Chr) construit la représentation en extension d'un historique sous forme d'une I_Chronique (respectivement X_chronique et D_Chronique) (Cf. figure 3.7b). En particulierΓces fonctions permettent de passer d'une représentation d'une chronique à une autre[4].

TYPE I_chronique (u, T) : [I_instantané (u, T)]
{ *soit* $C = [C_1, .., C_n]$ *de type I_chronique (u, T) :*
$\forall\ k \in [1..n-1],\ VT\ (C_k) < VT\ (C_{k+1})$ }
TYPE X_chronique (u, T) : [X_instantané (u, T)]
{ *soit* $C = [C_1, \ldots, C_n]$ *de type X_chronique (u, T) :*
$\forall\ k \in [1 \ldots n\text{-}1],\ VT\ (C_k) < VT\ (C_{k+1})$
Dans le cas d'une X_chronique canonique :
$\forall\ k \in [1 \ldots n\text{-}1],\ VT\ (C_k) < VT\ (C_{k+1}) \wedge$ *(EstContigu* $(VT\ (C_k),\ VT\ (C_{k+1}))$
$\Longrightarrow VS\ (C_k) \neq VS\ (C_{k+1}))$ }
TYPE D_chronique (u, T) : [D_instantané (u, T)]
{ *soit* $C = [C_1, \ldots, C_n]$ *de type D_chronique (u, T) :*
$\forall\ j, k \in [1 \ldots n],\ j \neq k \Longrightarrow VT\ (C_j) \cap VT\ (C_k) = \emptyset \wedge VS\ (C_j) \neq VS\ (C_k)$ }

(a) Sous-types du type Chronique

I_Chr : Historique (u, T) $\longrightarrow$ I_chronique (u, T)
X_Chr : Historique (u, T) $\longrightarrow$ X_chronique (u, T) { *canonique* }
D_Chr : Historique (u, T) $\longrightarrow$ D_chronique (u, T)

(b) Fonctions de construction de chroniques

FIG. 3.7 – *Fonctions et sous-types de Chronique*

3.2 Opérations de consultation

Le modèle comporte un ensemble d'opérations de haut niveau sur les historiques : elles permettent de raisonner sur les historiques indépendamment de leur représentation. Ces opérationsΓissues de divers contextesΓont été validées au fur et à mesure de l'étude et de la classification des requêtes les plus significatives [FCS96]. Nous illustrons ici deux types d'opérations : des extensions de l'algèbre relationnelle adaptées aux historiques et des itérateurs sur les séquences.

4. La conversion d'une I_chronique en une D_chronique (coalescing) est étudiée dans [BSS96].

Les requêtes portent sur l'exemple Oplate. Le fonctionnement de cette entreprise est décrit dans l'annexe A et la section A.7 comporte la modélisation des entités en TEMPOS. Les requêtes sont exprimées dans le langage OQL [ADF+94]Γétendu par les fonctions de TEMPOS[5]. L'expression de chaque requête est précédée de la formulation du type du résultat.

3.2.1 Projection et restriction temporelle

La projection d'un historique construit un historique de mêmes caractéristiques temporelles que l'historique donné. Le changement de type ne concerne que les valeurs structurelles. E étant un n-uplet de type T dont les noms de champs forment un ensemble N et X étant un sous-ensemble de NΓE[X] dénote la projection de E selon X. De même h étant de type Historique (u, T) où T est un n-uplet de typesΓla projection de h selon les noms XΓnotée h[X]Γest l'historique déduit de h en projetant chacun de ses instantanés selon X. Par ailleursΓles fonctions DomT et DomS (Cf. figure 3.5b) sont des projections particulières du fait que le résultat n'est pas un historique.

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

{ *type du résultat :* {<*numero : string, dateach : instant, ent :* { <*string, real*> } >} }
SELECT TUPLE (numero : c.Immat, dateach : c.DateAchat, ent : DomS(c.Entretien))
FROM c IN LesCamions

La *restriction temporelle* d'un historique h selon une propriété PΓnotée h ⌈ PΓest une chronique dont les instantanés sont ceux de h vérifiant P. Nous particularisons ici deux restrictions temporelles (Cf. figure 3.8a) : h $\lceil_{si}$ P est la chronique formée des I_instantanés de h dont les valeurs structurelles vérifient P ; h $\lceil_{en}$ F est la chronique formée des I_instantanés de h dont l'instant appartient à F.

Nous particularisons deux compositions des opérations de projection et de restriction : LesInstants et LesValeurs. LesInstants (h, P) (respectivement LesValeurs (h, P)) est l'ensemble des instants (respectivement des valeurs) des instantanés de h dont la valeur structurelle vérifie P (Cf. figure 3.8b).

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

{ *type du résultat :* {<*immat : string, ent : Chronique (mois, tuple(string, real))*>} }
SELECT TUPLE (immat : c.Immat,
ent : c.Entretien $\lceil_{si}$ λv • v.Nature ≠ 'Vidange'
FROM c IN LesCamions

5. Les notations utilisées sont décrites au début du document

TYPE Prédicat (T) : (T $\longrightarrow$ booléen)

_ $\lceil$ _ : Historique (u, T), Prédicat (I_instantané (u, T)) $\longrightarrow$ Chronique (u, T)
{ $h \lceil P = \{ <I, v> \mid <I, v> \in_i h \wedge P\ (<I, v>) \}$ }
_ $\lceil_{si}$ _ : Historique (u, T), Prédicat (T) $\longrightarrow$ Chronique (u, T)
{ $h \lceil_{si} F = \{ <I, v> \mid <I, v> \in_i h \wedge P\ (v) \}$ }
_ $\lceil_{en}$ _ : Historique (u, T), {Instant (u)} $\longrightarrow$ Chronique (u, T)
{ $h \lceil_{en} F = \{ <I, v> \mid <I, v> \in_i h \wedge I \in F \}$ }

(a) Restrictions temporelles d'historiques

LesInstants : Historique (u, T), Prédicat (T) $\longrightarrow$ { Instant (u) }
{ $LesInstants\ (H, P) = DomT\ (H \lceil_{si} P)$ }
LesValeurs : Historique (u, T), Prédicat (T) $\longrightarrow$ { T }
{ $LesValeurs\ (H, P) = DomS\ (H \lceil_{si} P)$ }

(b) Composition d'une projection et d'une restriction

FIG. 3.8 – *Projection et restriction temporelles*

_ $*_{\cap}$ _ : Historique (u, T1), Historique (u, T2) $\longrightarrow$ Chronique (u, <T1, T2>)
{ $h1 *_{\cap} h2 = \{ <I, <v1, v2>> \mid <I, v1> \in_i h1 \wedge <I, v2> \in_i h2 \}$ }

_ $*_{\cup}$ _ : Historique (u, T1), Historique (u, T2) $\longrightarrow$ Chronique (u, <T1, T2>)
{ $h1 *_{\cup} h2 = h1 *_{\cap} h2$
$\cup \{ <I, <v1, Nil>> \mid <I, v1> \in_i h1 \wedge I \in DomT\ (h1) \setminus DomT\ (h2) \}$
$\cup \{ <I, <Nil, v2>> \mid <I, v2> \in_i h2 \wedge I \in DomT\ (h2) \setminus DomT\ (h1) \}$ }

FIG. 3.9 – *Produits naturels temporels*

WHERE (c.Entretien $\lceil_{si}$ λv • v.Nature $\neq$ 'Vidange') $\neq \emptyset$

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

{ *type du résultat :* {<*type : string, prix : Chronique (jour, real)*>} }
SELECT TUPLE (type : b.Type,
prix : b.Prix $\lceil_{en}$ [@'1-1-1990'..@'1-12-1996'])
FROM b IN LesBouteilles
WHERE (b.Prix $\lceil_{en}$ [@'1-1-1990'..@'1-12-1996']) $\neq \emptyset$

Les requêtes **R.2** et **R.3** illustrent les situations où la même expression temporelle sert à décrire la sélection et la restriction. Dans le premier cas, la condition porte sur les valeurs temporelles (utilisation du $\lceil_{en}$) et dans le second cas, elle porte sur les valeurs structurelles (utilisation du $\lceil_{si}$). La requête suivante illustre une situation où les expressions temporelles spécifiant la restriction et la sélection sont indépendantes :

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

{ *type du résultat :* {<*nom : string, dir : Chronique (jour, string)*>} }
SELECT TUPLE (nom : c.Nom,
dir : c.Directeur $\lceil_{en}$ LesInstants (c.Coordonnees, λa • a.adresse = 'Paris'))
FROM c IN LesClients
WHERE DomT (c.Commandes)$^-$ $\leq$ (@'31-12-1994')
{ *L'historique résultat est vide pour les clients n'ayant pas été à Paris.* }

3.2.2 Produit naturel temporel

Le produit naturel temporel de deux historiques h1 et h2 de types Historique (u, T1) et Historique (u, T2) est une chronique de type Chronique (u, <T1, T2>) dont les instantanés sont formés à partir des instantanés donnés datés par le même instant. Nous distinguons deux opérations notées $*_\cap$ (produit interne) et $*_\cup$ (produit externe) selon que le domaine temporel du résultat est obtenu par intersection ou par union des domaines temporels des données (Cf. figure 3.9).

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

{ *type du résultat :* {*Instant (jour)*} }
SELECT LesInstants ((p.Production $*_\cap$z.Production), λ<pp, pz> • pp > pz)
FROM p IN LesBouteilles, z in LesBouteilles
WHERE p.Type = 'PlusSoif' and z.Type = 'Zébu'

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

L'interprétation de la requête formulée ici ne retient que les instantanés datés par les instants auxquels la production est connue (valeur non Nil) et auxquels le type de bouteille est en vente (valeur non Nil de l'attribut prix)Γque son prix soit connue ou non.

{ *type du résultat :* {<*type : string, prixprod : Chronique (jour,* <*integer, real*>*)*>} }
SELECT TUPLE (type : b.Type,
prixprod : ((b.Production $\lceil_{si}$ Est_non_Nil) $*_{\cap}$(b.Prix $\lceil_{si}$ Est_non_Nil))
$\lceil_{en}$ [@'1/1/1997', @'31/12/1997'])
FROM b IN Les Bouteilles

3.2.3 Itérateurs

La représentation en extension des historiques sous forme de chroniques permet de leur appliquer des itérateurs généraux définis sur les séquences [SFLM93]. Ceux-ci sont rappelés dans l'annexe C.

La fonction Application applique une fonctionΓélément par élément à tous les éléments d'une séquence. La fonction Sélection extrait d'une séquence tous les éléments vérifiant une propriété donnée. La fonction Agrégation applique une opération binaire de manière cumulative à tous les éléments d'une séquence à partir d'une valeur de base initiale.

Les fonctions LesCouplesΓLeDébut et LaFin permettent de formuler les requêtes impliquant un raisonnement sur la succession dans le temps en les appliquant à la représentation en extension des historiques sous forme de I_chroniques ou de X_chroniques. La fonction LesCouples construit les couples consécutifs d'une séquence. Les fonctions LeDébut et LaFin sont spécifiées par rapport au découpage d'une séquence en deux parties séparées par le premier élément (de gauche à droite) vérifiant une propriété donnée : LeDébut fournit les éléments de la séquence situés avant l'élément de séparation et LaFin fournit ceux situés après (y compris l'élément de séparation).

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

{ *type du résultat : integer* }
SELECT Agrégation (I_Chr (b.Production $\lceil_{en}$ LesInstants (b.Prix, λx • x > 5)),
0, λacc, i • acc + VS (i))
FROM b IN LesBouteilles
WHERE b.Type = 'PlusSoif'

La fonction Agrégation réalise le cumul acc de la production avec la valeur intitiale 0 et en accumulant les valeurs structurelles de chacun des I_instantanés i.

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

```
{ type du résultat : X_chronique (jour, tuple(string, real)) }
SELECT LaFin (c.Entretien, λx • VS (x).Nature = 'Vidange')
FROM c in LesCamions
WHERE c.Immat = '735 AOC 38'
```

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

```
{ type du résultat : [string] }
SELECT Application (
        Sélection (LesCouples (X_Hist (c.Coordonnees)),
                λ<i1, i2> • VS(i1).Adresse = 'Propriano' ∧ VS(i2).Adresse ≠ 'Propriano'),
        λ<i1, i2> • VS (i2.Adresse)
FROM c IN LesClients
WHERE c.Nom = 'Alimentation du maquis'
```

La requête est basée sur la sélection dans l'historique des coordonnées des couples consécutifs de X_instantanés dont la première adresse est Propriano et la seconde est différente de Propriano. La fonction Application permet d'obtenir les adresses attendues.

3.3 Opérations de mise à jour

3.3.1 Mise à jour des historiques

Les opérations de mise à jour sont de différentes natures : ajout modification ou suppression d'une valeur. Ces opérations peuvent concerner le domaine temporel et/ou la chronique. Nous présentons ici les différents cas possibles :

- Ajout d'un instantané $< i, v >$:
  - $i \in DomEff$: dans ce cas il s'agit d'une correction de la valeur structurelle de l'instantané existant à l'instant i.
  - $i \in DomPot$: il s'agit de l'ajout de l'instantané $< i, v >$ à la chronique ; le domaine temporel reste inchangé (Cf. cas 1 de la figure 3.10).
  - $i \notin DomT$: ici l'ajout de l'instantané ajoute également i au domaine temporel. L'opération peut conduire à une valeur isolée ou non selon que i jouxte le domaine temporel ou non. Pour éviter de laisser une valeur isolée il est possible de compléter le domaine temporel à droite et/ou à gauche du point d'insertion comme le montre le deuxième cas de la figure 3.10.

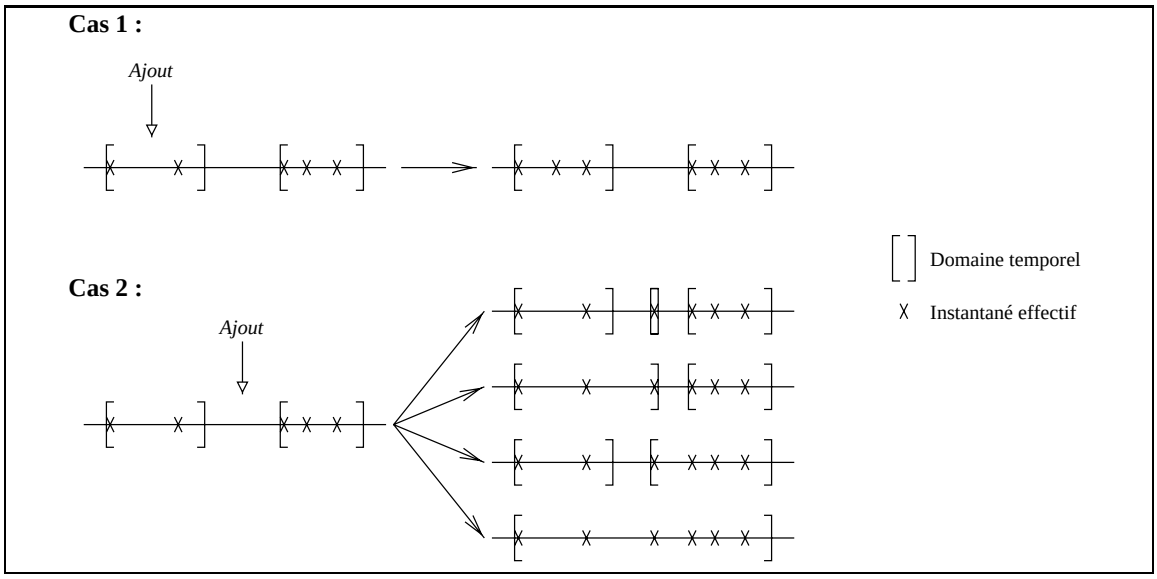


FIG. 3.10 – *Ajout d'un instantané à un historique en compréhension*

– Suppression d'un instantané $< \mathsf{i}, \mathsf{v} >$: l'instantané est supprimé de la chronique. La question qui se pose pour cette opération est de savoir si :
  – i reste dans DomT et devient un point potentiel (cas 1 de la figure 3.11).
  – i est également supprimé de DomT (cas 2 de la figure 3.11).

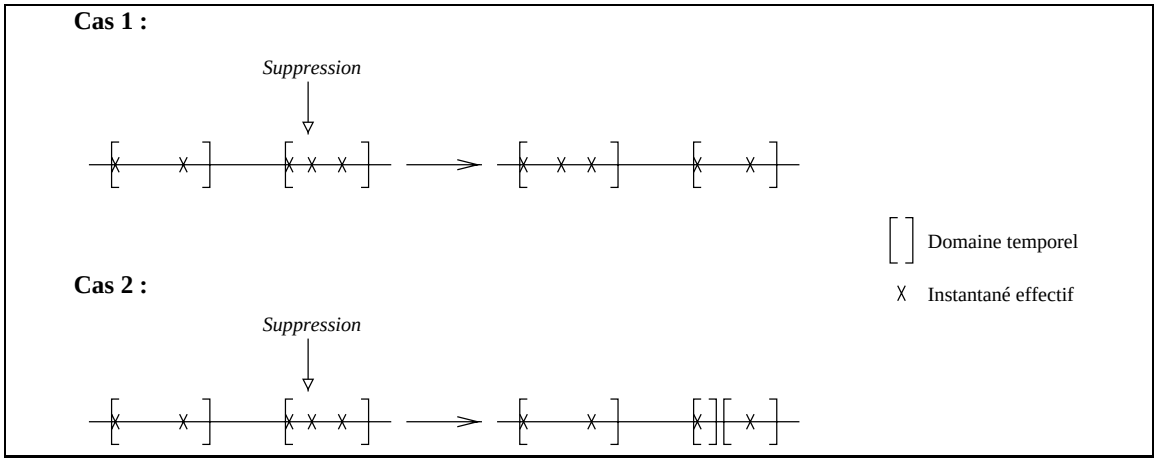


FIG. 3.11 – *Suppression d'un instantané d'un historique en compréhension*

3.3.2 Opérations élementaires

Nous définissons ici les opérations élémentaires de mises à jour qui s'appliquent sur le type Chronique et sur le type Hist.

Les opérations sur le type Chronique sont définies dans la figure 3.12. Elles correspondent à l'ajout ou la suppression d'un instantané à une chronique. Pour la suppression

d'un instantanéΓla donnée de l'instant est suffisante. Ces opérations sont étendues à l'ajout (ou la suppression) d'un ensemble d'instantanés.

La figure 3.13 présente les opérations relatives au type Hist. Ceux-ci étant composés d'une chronique et d'un domaine temporelΓil y a des opérations sur l'un ou sur l'autre : ajout ($+_T$) ou suppression ($-_T$) d'instants au domaine temporelΓajout ($+_C$) ou suppression ($-_C$) d'instantanés de la chronique. Ces fonctions sont généralisées à un ensemble d'instants ou à un ensemble d'instantanésΓselon le cas.

La description de ces fonctions appelle quelques remarques :

- Pour définir les opérationsΓnous raisonnons sur le couple $< C, P >$Γoù C est la chronique (qui fixe le domaine effectif) et P le domaine potentiel de l'historique considéré.

- Le fait de supprimer un instantané de la chronique fait passer un point effectif du domaine en un point potentiel. Le cas échéantΓla suppression du point potentiel doit se faire par suppression d'un instant du domaine temporel.

- Nous avons appelé *correction* l'opération qui modifie la valeur structurelle d'un instantané existant.

Soient C une chroniqueΓi un instantΓet $< i, v >$ un instantané.

- $C + < i, v > =$
  $\{ < i', v' > \mid < i', v' > \in C \wedge i' \neq i \} \cup \{ < i, v > \}$
- $C - i =$
  $\{ < i', v' > \mid < i', v' > \in C \wedge i' \neq i \}$

FIG. 3.12 – *Opérations de construction sur le type* Chronique

Soient C = ChronEff(H) et P = DomPot (H)Γ
H' le résultat de l'opération et C' = ChronEff(H')ΓP' = DomPot (H').

Opérations portant sur le domaine temporel :

- $H +_T i : < C', P' > =$
  $\hookrightarrow i \in DomT(H) : < C, P >$
  $\hookrightarrow i \notin DomT(H) : < C, P + i >$
- $H -_T i : < C', P' > =$
  $\hookrightarrow i \in DomPot(H) : < C, P - i >$
  $\hookrightarrow i \in DomEff(H) : < C - i, P >$
  $\hookrightarrow i \notin DomT(H) : < C, P >$

Opérations portant sur la chronique :

- $H +_C < i, v > : < C', P' > =$
  $\hookrightarrow i \in DomEff(H) : < C + < i, v >, P > \{ \textit{correction} \}$
  $\hookrightarrow i \in DomPot(H) : < C + < i, v >, P - i >$
  $\hookrightarrow i \notin DomT(H) : < C + < i, v >, P >$
- $H -_C i : < C', P' > =$
  $\hookrightarrow i \in DomEff(H) : < C - i, P + i >$
  $\hookrightarrow i \notin DomEff(H) : < C, P >$

FIG. 3.13 – *Opérations de construction sur le type* Hist

3.3.3 Opérations sur les attributs historiques

Nous définissons maintenant les opérations de mises à jour sur les attributs historiques. Nous définissons deux séries d'opérateurs suivant qu'il s'agit d'un attribut historique régulier (Cf. figure 3.14) ou d'un historique discret en escalier ou interpolé (Cf. figure 3.15).

Les opérations sur les historiques réguliers ne peuvent affecter directement le domaine temporel. Inversement les opérations sur les historiques discrets en escalier ou interpolés peuvent affecter le domaine temporel ou la chronique effective.

Dans tous les cas il s'agit d'opérations pour :

- Ajout d'un instantané
- Correction de la valeur structurelle d'un instantané
- Correction de la valeur temporelle d'un instantané
- Suppression d'un instantané
- Extension du domaine temporel
- Restriction du domaine temporel

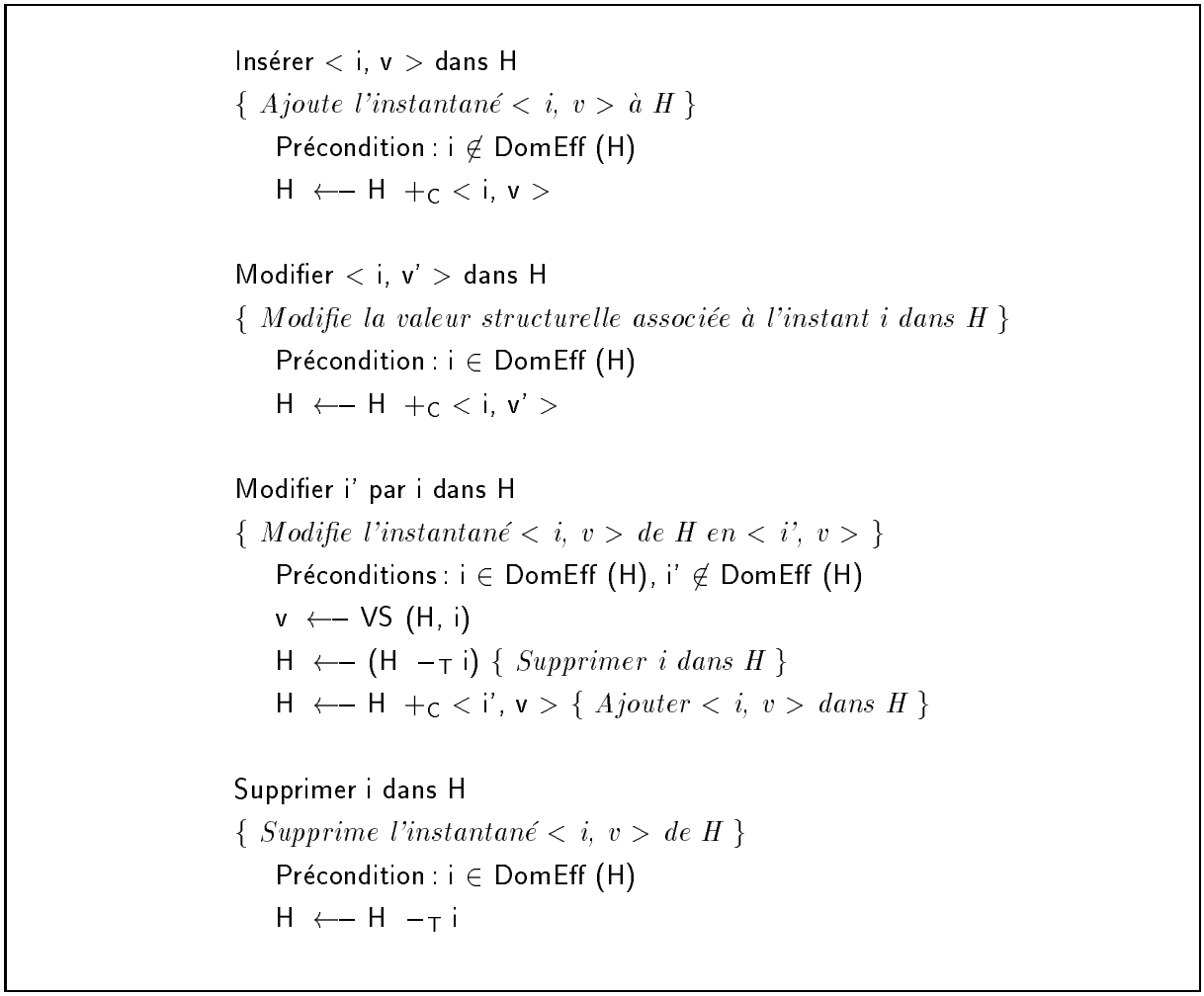


FIG. 3.14 – *Opérations de mise à jour des attributs historiques réguliers*

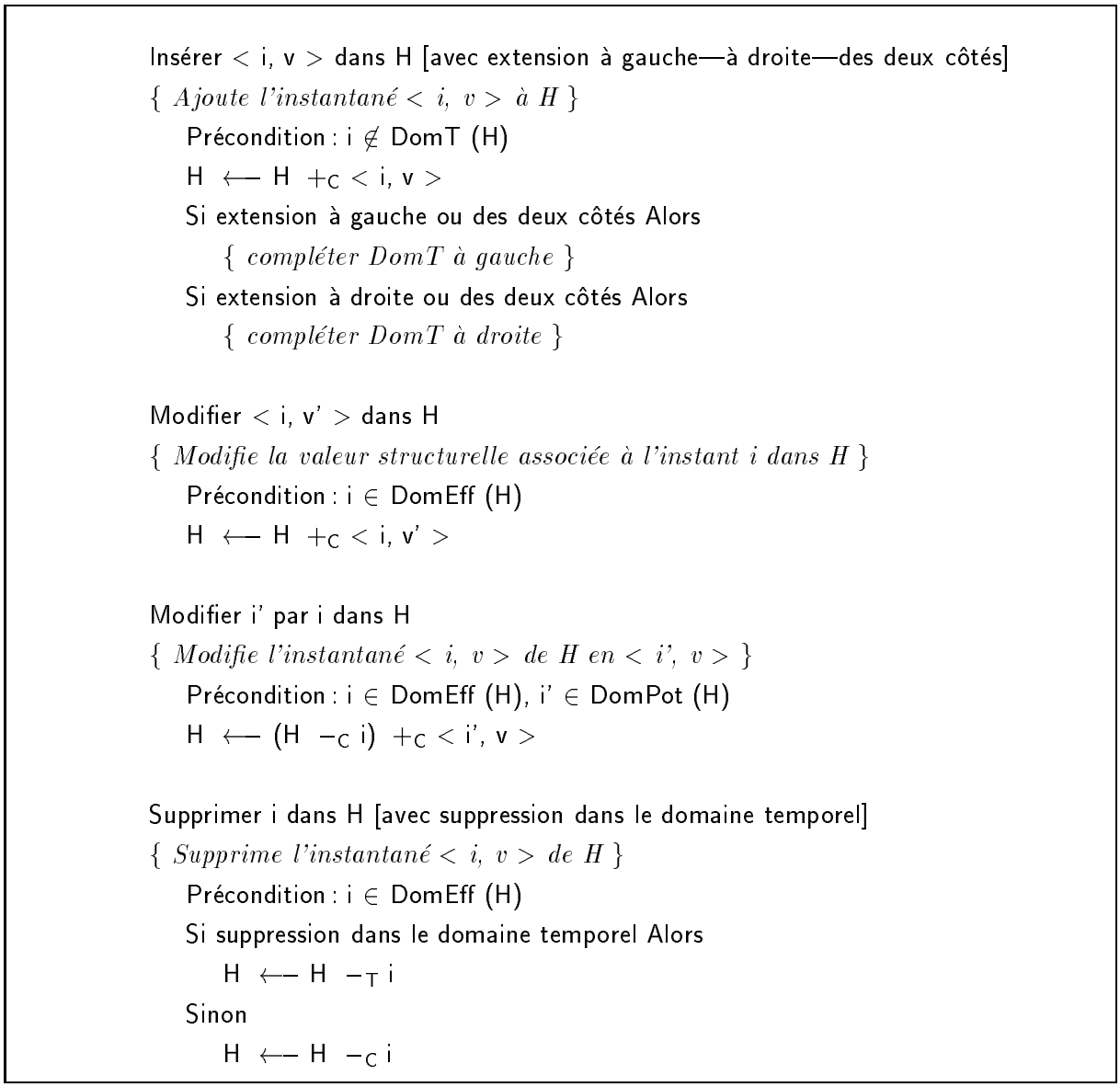


FIG. 3.15 – *Opérations de mise à jour des attributs historiques discrets, en escalier ou interpolés*

Chapitre 4

Tempos : Expérimentation

Nous présentons dans ce chapitre le prototype du modèle TEMPOS en cours de réalisation [Can96cΓCan96bΓCan96aΓDum96ΓCD97] au dessus du SGBD O_2 [Tec95ΓAC93].

4.1 Architecture

Le prototype du modèle TEMPOS est constitué d'une bibliothèque de classes pour le SGBD O_2 et d'un préprocesseur (Cf. figure 4.1).

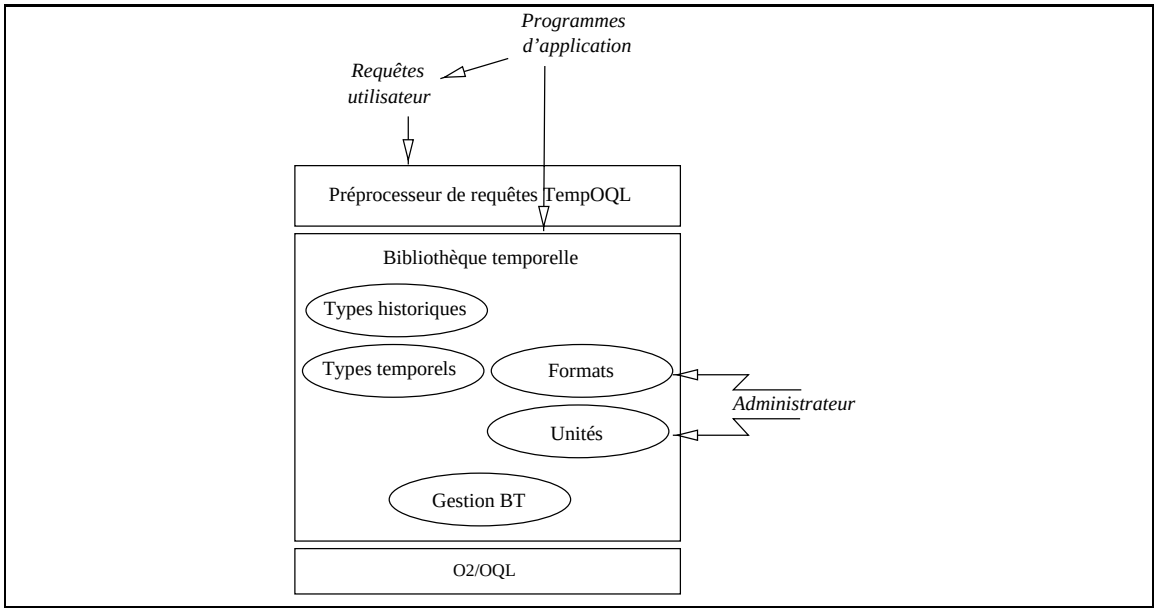


FIG. 4.1 – *Architecture du prototype*

La bibliothèque temporelle comporte cinq groupes de classes pour la description des unités de temps et des formatsΓles types temporelsΓles types historiques et enfin pour le fonctionnement général de la bibliothèque.

Les quatres premiers groupes sont détaillées dans les sections suivantes. Les classes relatives au fonctionnement de la bibliothèque permettent de :

- Effectuer les différentes initialisations nécessaires.
- Faire des traitements nécessaires à toutes les autres classes.
- Gérer les erreurs.

Quant au préprocesseur, il transforme les requêtes TempOQL en OQL standard. TempOQL permet la manipulation de valeurs temporelles au même titre que les autres types de base. Pour cela, il utilise évidement les fonctionnalités offertes par la bibliothèque temporelle.

Les fonctionnalités temporelles sont accessibles par les programmes d'application au travers de requêtes temporelle en TempOQL ou bien par des appels directs de méthodes, en O_2C par exemple.

> **Remarque :** Les classes relatives aux formats et le préprocesseur TempOQL ont été développés par Marlon Dumas pendant son stage de magistère [Dum96].

4.2 Représentation du temps

4.2.1 Types temporels

Un premier groupe de classes correspond aux types des valeurs temporelles (durées, instants et séquences d'instants) (Cf. figure 4.2).

A chaque type temporel défini dans le modèle TEMPOS est associé une classe. Pour chacune d'elles, les méthodes définies correspondent aux opérations spécifiées pour son type. Cette partie du prototype n'est pas modifiable par l'administrateur.

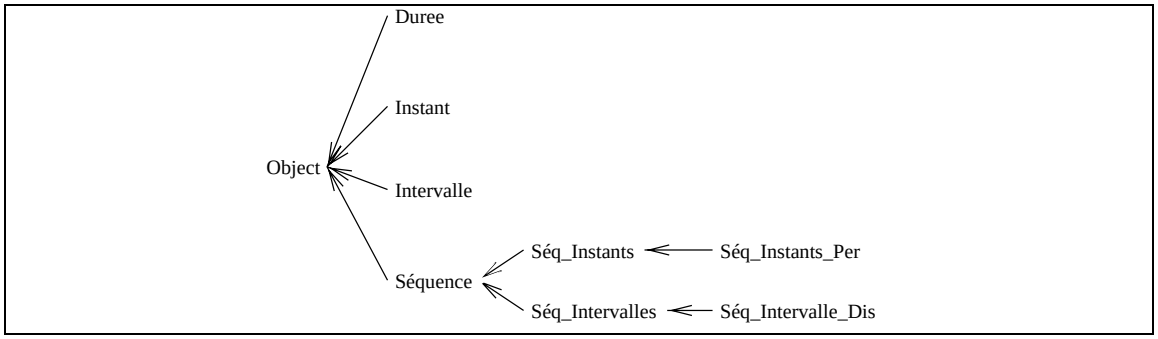


FIG. 4.2 – *Classes relatives aux types temporels*

4.2.2 Unités

Un premier groupe de classes concerne la description des unités d'observation du temps, des couples d'unités (pour ce qui est lié aux conversions entre unités) et des systèmes d'unités (Cf. figure 4.3) :

- La classe UT correspond à la description de chaque unité temporelle : son nom, son origine, etc. Les unités régulières sont isolées dans une sous-classe particulière UT_Reg. Ces deux classes sont virtuelles ; chaque unité correspond à une sous-classe (par exemple, UT_Jour). Sont rattachées à ces classes, les méthodes spécifiques à chaque unité (comme par exemple, celle permettant de savoir si une année est bissextile) et des méthodes générales (comme la relation d'ordre $\prec$ sur les unités).

- La classe CUT (resp. CUT_Reg) correspond aux couples d'unités (resp. réguliers). Il s'agit également de classes virtuelles, chaque couple particulier demandant une classe particulière (par exemple, CUT_An_Jour).
  Les méthodes de ces classes décrivent les fonctions d'approximation et d'expansion (Cf. section 2.1.3).

- la classe SUT (resp. SUT_Reg) correspond aux systèmes d'unités (resp. réguliers). Les méthodes de ces classes décrivent les fonctions de réduction d'une forme multigranulaire en une forme canonique et les fonctions de décomposition d'une forme canonique en une forme multigranulaire (Cf. section 2.3.2).

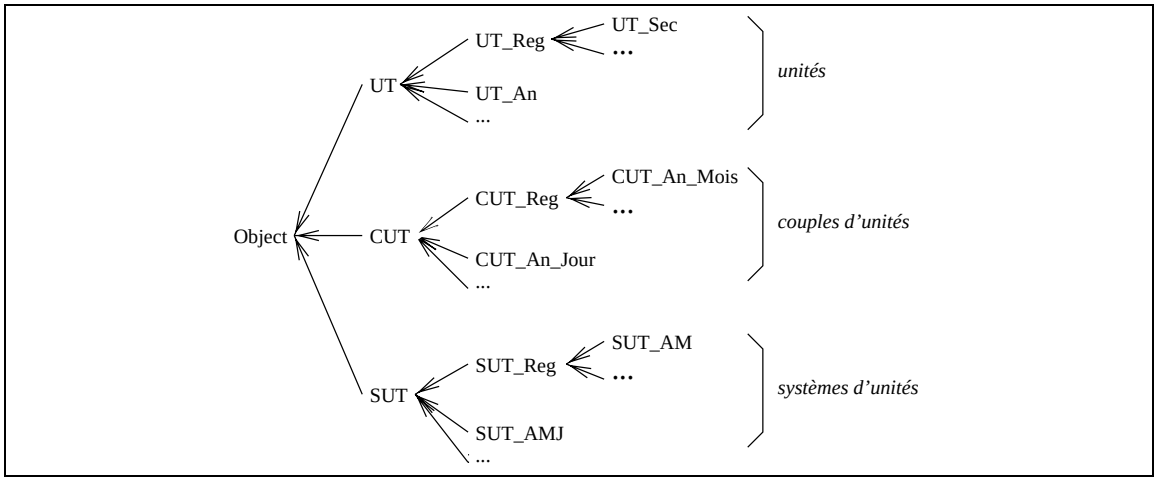


FIG. 4.3 – *Classes relatives à la description des unités temporelles*

Lors de l'initialisation de la bibliothèque, il est nécessaire de créer une instance de chaque classe correspondant à une unité (par exemple, UT_Sec, ... UT_An), à un couple d'unités (par exemple, CUT_An_Mois) ou à un système d'unités (par exemple, SUT_AMJ). Un nom (racine de persistance) est défini pour chacune de ses instances. Ainsi, toutes les

opérations sur les unités temporelles sont en fait des appels de méthodes de ces objets nommés.

4.2.3 Formats

Un troisième groupe de classes de la bibliothèque contient les classes relatives à la gestion des formats et des systèmes de formats.

Les notions de format et de système de formats introduites dans la section ?? sont modélisées respectivement par les types *format* et *système de formats*. A chaque type temporel correspond un sous-type de *format* et un sous-type de *système de formats*. AinsiΓ il existe 5 sous-types de *format* correspondant aux formats d'instantsΓaux formats de duréesΓde durées signéesΓde durées relatives et de durées relatives regulières.

A chacun de ces types correspond une classe. La hiérarchie de ces classes est calquée sur celle des classes associées aux types temporels de base comme le montre la figure 4.4. Ces classes possèdentΓentre autresΓles méthodes pour construire une valeur temporelle à partir d'une forme externe et vice-versa.

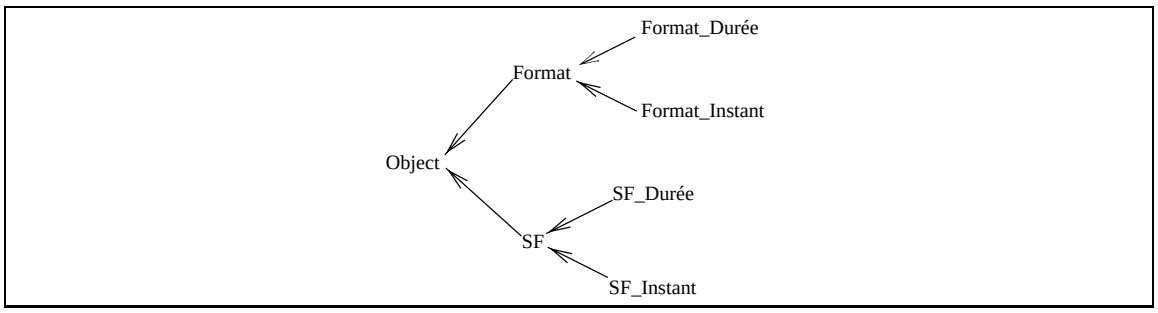


FIG. 4.4 – *Classes relatives aux formats*

4.3 Historiques

Les classes relatives aux historiques sont en cours d'implantation. Nous présentons ici la structure générale retenue et les principaux problèmes d'implantation des historiques.

4.3.1 Types historiques

Comme pour les types temporelsΓune classe est créée pour chaque type de TEMPOS. La hiérarchie obtenue est illustrée par la figure 4.5.

Les fonctions définies sur les types historiques sont implantées par des méthodes dans les classes correspondantes.

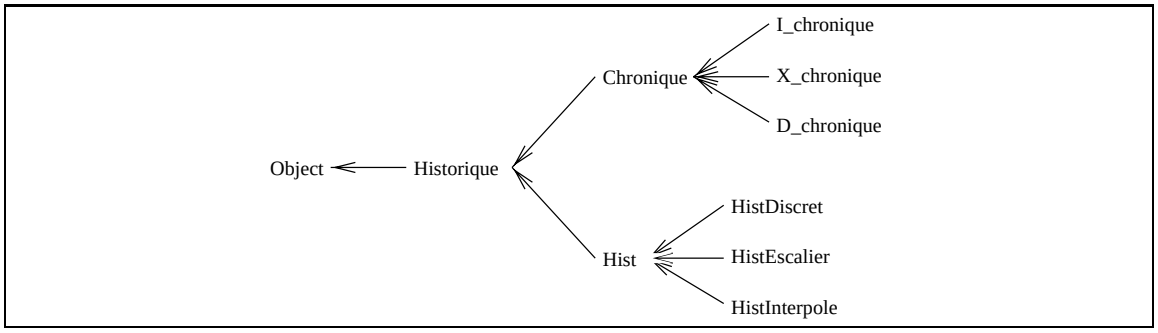


FIG. 4.5 – *Classes relatives aux types historiques*

4.3.2 Problèmes d'implantation

Plusieurs problèmes se posent pour l'implantation des types historiques. Certains d'entre eux sont dus au SGBD et d'autres à TEMPOS :

- O_2 ne gère pas les classes paramétrées or les types historiques correspondent à des types paramétrés par l'unité d'observation et la valeur structurelle.
  En ce qui concerne l'unité d'observation cela ne pose pas de problème car l'unité d'observation est traitée comme un attribut des historiques comme cela a été fait pour les types temporels.
  Par contre pour la valeur structurelle il est important que le type soit réellement paramétré. Cela permet les appels successifs de méthodes et en particulier sur les types des valeurs structurelles que l'on peut obtenir à partir des manipulations sur les historiques. Pour résoudre ce problème il faut utiliser l'héritage comme le montre la figure 4.6. Par exemple dans la classe HistDiscret le type de la valeur structurelle est Object et dans la sous classe HD_Commande qui correspond à un historique discret de commandes le type de la valeur structurelle est Commande.
  Ceci ne résoud pas le problème pour les opérations (comme le produit naturel) qui créeent dynamiquement des historiques dont le nouveau type ne correspond généralement pas à une classe existante.

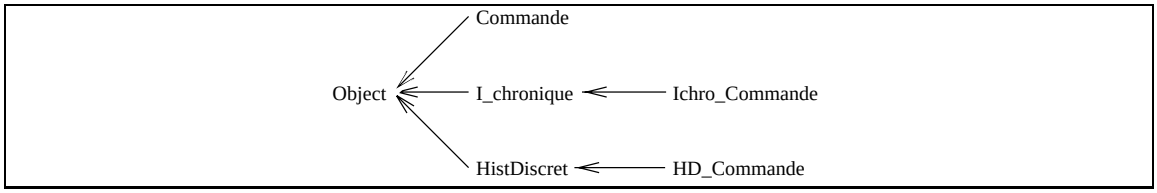


FIG. 4.6 – *Exemple d'utilisation de l'héritage pour les classes paramétrées*

- Les opérations de restriction temporelles sur les historiques proposées par TEMPOS ont un prédicat en paramètre. Celui-ci doit être exprimé par une lambda expression. Pour

ne pas implanter un interpréteur de lambda expressionsΓil faut mettre en évidence la fonctions les plus courantes et restreindre les prédicats possibles à une liste prédéfinie. Parmi les prédicats les plus courantsΓon trouve ceux définis dans les langages comme SQLΓc'est à dire l'égalitéΓla relation d'ordre.
De mêmeΓpour l'utilisation des fonctions génrales sur les séquencesΓil faut passer des fonctions en paramètre. Là encoreΓil faut trouver celles qui sont les plus utilisées. Par exempleΓpour les réductionsΓcela correspond aux opérateurs minΓmaxΓetc. de SQL.

4.4 Paramétrisation de la bibliothèque temporelle

La modélisation du temps proposée par TEMPOS est basée sur un système extensible d'unités d'observation du temps. Cette extensibilité est préservée par le prototypeΓet les sections suivantes montrent comment définir des nouvelles unités et de nouveaux formats.

Pour celaΓil faut intervenir dans la bibliothèque temporelle et définir de nouvelles classes. Ce n'est pas un travail très convivial et très simpleΓaussiΓle développement d'outils d'administration doit être étudié.

4.4.1 Définition de nouvelles unités

Pour rajouter de nouvelles unitésΓl'utilisateur doit fournir les caractéristiques propres à l'unité ajoutée et les fonctions de conversions avec les unités qui lui sont comparables. Pour celaΓil doit :

- Décrire les caractéristiques propres à la nouvelle unité en créant une sous-classe de UT.
- Donner les fonctions de conversions entre cette nouvelle unité et les unités auxquelles elle est comparable en créant des sous-classes de CUT et en y redéfinissant les méthodes liées à l'expansion et à l'approximation. Les méthodes correspondant aux conversions peuvent ainsi être implantées de la façon la plus efficace possible en fonction des caractéristiques propres à chaque couple d'unités.

Les unités et les types temporels étant clairement séparésΓune unité rajoutée est directement utilisable pour l'expression d'une valeur temporelleΓet ne modifie en rien l'expression des valeurs n'utilisant pas cette nouvelle unité.

4.4.2 Définition de nouveaux systèmes d'unités

Contrairement à la création de nouvelles unitésΓl'utilisateur peut définir de nouveaux systèmes sans créer de classe. Il suffit de créer une instance de la classe SUT. Cela peut donc être fait dynamiquement par une application. C'est un point particulièrement important car les systèmes d'unités peuvent être extrêmement nombreux à partir des unités définies.

Lors de la création d'une instanceΓil suffit d'affecter à un attribut donné la liste des unités qui composent le nouveau système. Dans ce casΓles fonctions de réduction et de décompositionΓutilisées pour passer des formes multigranulaires à la forme irréductibleΓ utilise un algorithme général prédéfini au niveau de la classe SUT.

Pour optimiser les calculsΓl'utilisateur a la possibilité de définir une sous classe spécifique à un système d'unités donnéΓet il peut alors redéfinir les méthodes liées à la réduction et à la décomposition.

AinsiΓbien que l'ensemble des unités et des systèmes d'unités soit extensibleΓles performances ne sont pas dégradées puisqu'il est possible d'utiliser des algorithmes adaptés à chaque cas.

4.4.3 Définition de nouveaux formats

EnfinΓL'utilisateur peut créer de nouveaux formats et de nouveaux système de formats.

Dans ce casΓla création se fait dynamiquement par les applications. Pour les formatsΓil suffit d'indiquer le système d'unités concernéΓla nature des valeurs (textuelleΓnumérique (chiffres arabes ou romains)) et la permutation à utiliser (Cf. section 2.5.2).

Pour la création des systèmes d'unitésΓle principe est le mêmeΓmais la tâche peut être plus complexe car l'utilisateur doit vérifier que le système qu'il crée (ou qu'il modifie) ne comporte pas des formats incompatibles (comme le format européen et le format américain de dates).

4.5 Préprocesseur TempOQL

4.5.1 Réalisation du préprocesseur

Le langage TempOQL a été implanté au dessus de O_2/OQL au moyen d'un préprocesseur qui traduit des requêtes **TempOQL** vers des requêtes OQL enrichi des classes et noms persistants fournis par la bibliothèque temporelle. Tout comme le langage TempOQLΓle préprocesseur est paramétré par des systèmes de formats qui déterminent l'interprétation des constantes temporelles de base rencontrées dans les requêtes.

L'analyseur lexico-syntaxique du préprocesseur a été implanté en C à l'aide des générateurs d'analyseurs lexicaux et syntaxiques traditionnels *Lex* et *Yacc* [ASU86]. L'analyseur sémantique de son côté a dû être implanté en O_2C du fait des appels aux méthodes de la bibliothèque temporelle. L'intégration de ces deux composants a été faite en utilisant l'*interface C pour* O_2 [Tec95] dans un premier tempsΓpuisΓpour des questions de performanceΓen important l'analyseur syntaxique dans O_2 en tant qu'une bibliothèque C.

L'extension des opérateurs arithmétiques et booléens aux valeurs temporelles suppose une connaissance du type des expressions manipulées. L'analyseur sémantique du pré-

processeur comporte donc un sous-système d'inférence de type. Ce sous-système a été developpé grâce aux fonctionnalités du *Méta-Schéma* de O_2 [Tec95].

Le préprocesseur de requêtes est accessible soit en mode embarqué par l'intermédiaire d'une fonction qui traduit une chaîne de caractères contenant une requête TempOQL en une autre contenant la requête OQL équivalente, soit en mode interactif grâce à une application O_2. Cette application est constituée d'une boucle qui saisit des requêtes TempOQL, les traduit en OQL à l'aide du préprocesseur, et les soumet à l'interpréteur OQL du système O_2. L'application offre aussi un manuel en ligne décrivant les fonctionnalités propres au langage TempOQL et gère les préférences utilisateurs en ce qui concerne les systèmes de formats utilisés lors de l'interprétation des constantes temporelles.

4.5.2 Exemples d'utilisation de TempOQL

Nous décrivons maintenant des expressions TempOQL, dans le cadre de l'application Oplate (Cf. annexe A), montrant quelques caractéristiques de TempOQL.

```
SELECT C.Immat
FROM C IN LesCamions
WHERE C.DateAchat BELONGS_TO [@'1/6/1996'..@'31/5/1997']
```

Cette requête sélectionne les numéros de plaques d'immatriculation des camions achetés entre le 1/6/1996 et le 31/5/1997. Elle illustre :

- La manipulation de constantes de type instant, marquées par le symbole @ : @'1/6/1996'.
- La construction d'un intervalle, avec les symboles [..].
- L'utilisation d'une fonction (BELONGS_TO) entre un instant et un intervalle, manipulée comme un opérateur OQL.

La requête OQL équivalente, qui contient les appels à la bibliothèque temporelle, est la suivante :

```
SELECT C.Immat
FROM C IN LesCamions
WHERE C.DateAchat->I_Appartient_X (Intervalle()->X_Cons_I_I (
                                   Instant()->I_Cons_FE (list(1996.0, 6.0, 1.0), SUT_AMJ),
                                   Instant()->I_Cons_FE (list(1997.0, 5.0, 31.0), SUT_AMJ)))
```

{ *Les méthodes utilisées de la bibliothèque sont :*
- *I_Appartient_X, prédicat qui indique si un instant appartient à un intervalle,*
- *X_Cons_I_I, qui construit un intervalle à partir de deux instants,*
- *I_Cons_FE, qui construit un instant à partir de sa forme externe.* }

{ *L'appel à Instant() et Intervalle() permet de créer des objets temporaires du type correspondant* }

La requête suivante illustre les conversions d'unités dans le cas des instants. L'opération de changement d'unité en TempOQL est noté !. La requête donne l'année d'achat du camion immatriculé 735 AOC 38 :

```
SELECT C.DateAchat ! Année
FROM C IN LesCamions
WHERE C.Immat = '735 AOC 38'
```

Sa traduction en OQL est :

```
SELECT C.DateAchat– >I_Conv (UT_An)– >X_Binf
FROM C IN LesCamions
WHERE C.Immat = '735 AOC 38'
```

{ *Les méthodes utilisées de la bibliothèque sont :*
– *I_Conv, qui convertit un instant dans une unité donnée (le résultat est un intervalle),*
– *X_Binf , qui retourne la borne inférieure d'un intervalle.* }

Conclusion

Bilan

Après un état de l'artΓdans le premier chapitreΓqui a introduit les principaux concepts relatifs au domaine des bases de données temporellesΓet qui a également présenté quelques travaux significatifs dans le domaineΓnous avons introduit le modèle TEMPOS.

Ce modèle est composé d'un modèle de représentation du temps et un modèle d'historiques. Pour la représentation du tempsΓTEMPOS offre une vision multigranulaire basée sur le découpage du temps en unités d'observation du temps. Celles-ci unifie les concepts habituels de repères calendairesΓutilisés pour exprimer des datesΓet d'unités de tempsΓutilisées pour la mesure de durées. La relation "est plus fine"Γentre les unités d'observationΓ permet de définir toutes les fonctions relatives aux conversions. Ce système d'unités est extensibleΓet l'utilisateur (ou l'administrateur) peut ajouter les unités spécifiquesΓnécessaires à ses applications. Les types temporels proposés par TEMPOS sont très riches : des types simplesΓcomme les instants et les duréesΓaux plus complexes comme les séquences d'instants et les calendriers. Ils sont tous munis de multiples opérations (constructeursΓ relations d'ordreΓprédicatsΓetc.). Les types temporels sont indépendants des unitésΓet les opérations entre valeurs de granularités différentes sont définies uniquement dans les cas valides.

En ce qui concerne le modèle d'historiquesΓTEMPOS permet de fixer le statut d'évolution des attributs : constantΓfugitif ou historique. Les différents types d'historiques permettent une interprétation adaptée à la nature des phénomènes : discretΓen escalierΓ interpolé. Les représentationsΓen extension ou en intentionΓpermettent de stocker les valeurs signicatives. Ces représentations sont basées sur les chroniquesΓqui correspondent à des séquences d'instantanés. Les instantanés sont les valeurs estampillées des attributs. Les historiques sont indépendants des unités d'observation et les estampilles peuvent être de n'importe quelle granularité. La donnée d'un domaine temporel etΓéventuellementΓd'une fonction d'interpolation permet de déduire les valeurs de l'attribut entre deux valeurs saisies. La vision fonctionnelle des historiques permet de raisonner en faisant abstraction de la représentation utilisée.

Pour l'interrogation des historiqueΓTEMPOS propose un jeu d'opérateurs de haut niveauxΓissus de l'extension temporelle des opérateurs de l'algèbre relationnelle et des

fonctions générales de manipulation de séquences. Ces opérateurs sont définis indépendamment de la représentation des historiques. Ceci permet en particulier de choisir la représentation la plus adaptée à chaque requête, ce qui donne un style d'expression uniforme pour toutes les requêtes.

L'expérimentation en cours au dessus du SGBD O_2 a déjà permis de valider de nombreux points du modèle TEMPOS. De même, l'application de TEMPOS au cas des séries chronologiques a permis de montrer que le modèle peut répondre à des besoins précis des applications.

Perspectives

Ce travail correspond à une première étape significative de l'axe "Bases de données temporelles" du projet STORM.

Aussi, les poursuites envisageables sont nombreuses :

- Au niveau du modèle du temps :
  - Extension des instants (valeurs temporelles absolues) par la prise en compte d'instants relatifs (valeurs temporelles relatives). Il s'agit ici d'aller vers une représentation symbolique (et non plus numérique) du temps et d'intégrer la notion d'*événements temporels*. Des travaux dans ce sens ont déjà été menés dans le groupe STORM [Poi96, Fay97], pour intégrer des événements temporels dans les bases de données actives.
  - Une autre extension est la prise en compte des informations imprécises. Un fait n'est plus situé par un instant précis mais par une période floue : entre le 1er et le 15 juillet, après le 1er juillet, et avant le départ en vacances, etc.
- Au niveau du modèle d'historiques :
  - Etendre le modèle pour prendre en compte les historiques d'objets ou de collections d'objets.
  - Etendre les opérateurs relatifs à l'interrogation des historiques. En particulier, il est envisageable d'intégrer des opérateurs des logiques temporelles.
  - Etudier en détail le cas du bitemporel. Le temps de transaction répond à des besoins spécifiques, en particulier pour l'audit des opérations effectuées sur une base de données. Son traitement n'est pas similaire à celui du temps de validité et des opérateurs spécifiques sont à définir.
  - Définir un langage de définition des données et un langage de manipulation des données. Pour les opérations déjà définies dans le modèle, il faut masquer l'utilisation des lambda expressions pour les utilisateurs.

- Prendre en compte la gestion de contraintes d'intégrités temporelles. Les contraintes envisageables sont très nombreuses lorsque l'on garde l'historique des modifications. Les corrections effectuées dans des états passés de la base engendrent également de nouveaux problèmes. Des travaux ont déjà commencé dans ce sens dans le groupe STORM [DFG+96ΓDum97].
- Etudier les structures physiques et l'indexation des attributs historiques. L'historisation des attributs implique une augmentation du nombre d'informations à traiterΓ il est vital pour un SGBD temporelΓde respecter un temps de réponse acceptable.
- Etudier l'optimisation des requêtes. L'implantation des opérateurs sur les historiques doit être optimisée en fonction des diverses représentation des chroniques.

Par rapport au prototypeΓil est nécessaire de fournir des outils d'administrationΓfacilitant en particulier la définition de nouvelles unités et de nouveaux formats. La portabilité de l'implantation doit être étudiéeΓavec l'objectif de réaliser un serveur temporel adaptable à d'autres SGBD.

Il est également nécessaire de confronter le modèle proposé à d'autres application afin de le valider etΓéventuellementΓde le compléter.

L'application aux Systèmes d'Informations Géographiques (SIG) dans le cadre de coopérations entre l'axe temporel du projet STORM et d'autres équipes spécialisées dans les SIG ou les bases de données spatiales estΓà ce titreΓparticulièrement prometteuse. De plusΓc'est le cadre idéal pour étudier les similarités éventuelles entre bases de données temporelles et bases de données spatialesΓque ce soit au niveau de la modélisationΓde la représentation ou de l'interrogation.

EnfinΓparmi les perspectives essentielles à la proposition du modèle TEMPOSΓil y a l'étude de l'impact de la dimension temporelle au niveau de la modélisation des applications. Il faut proposer des éléments permettant la conception et la spécification d'applications manipulant des données temporelles dans les méthodes classiques. Pour celaΓ l'analyse faiteΓdans la premier chapitreΓdes notions clées du domaineΓest un point de départ intéressant.

Annexe A

Exemple : l'entreprise Oplate

Nous présentons ici une application qui sert à illustrer les différents points abordés au cours de ce document.

Dans un premier temps, nous exposons le contexte de cet exemple, et nous en donnons une modélisation. Il faut bien noter que cet exemple, basé sur un cas réel, a été simplifié pour que sa complexité ne vienne pas s'ajouter à celles des notions que nous voulons illustrer.

Ensuite, nous donnons sa modélisation ainsi que l'expression de quelques requêtes significatives dans différents contextes : SGBD relationnels temporels (TSQL 2, TempSQL) ou non temporel (Oracle), SGBD à objets temporel (TOOSQL) ou non temporel (O_2), et enfin dans le modèle TEMPOS. Dans un premier temps, la modélisation proposée est mono-temporelle (suivant le temps de validité). Une dernière section illustre rapidement le cas du bitemporel.

A.1 Position du problème

Nous considérons le cas d'une usine d'exploitation d'une source, qui produit des bouteilles d'eau minérale.

L'usine est installée sur le site d'une source et l'eau est immédiatement embouteillée sur une chaîne de production. Les bouteilles produites sont stockées sur place. L'usine livre elle-même ses clients à l'aide de ses propres camions.

La figure A.1 propose une modélisation OMT de cette application. Les attributs historisés sont marqués en italiques. Le lien indiqué entre Source et Bouteille n'est pas représenté par la suite car nous nous limitons à l'exploitation d'une seule source ; ce lien est implicite.

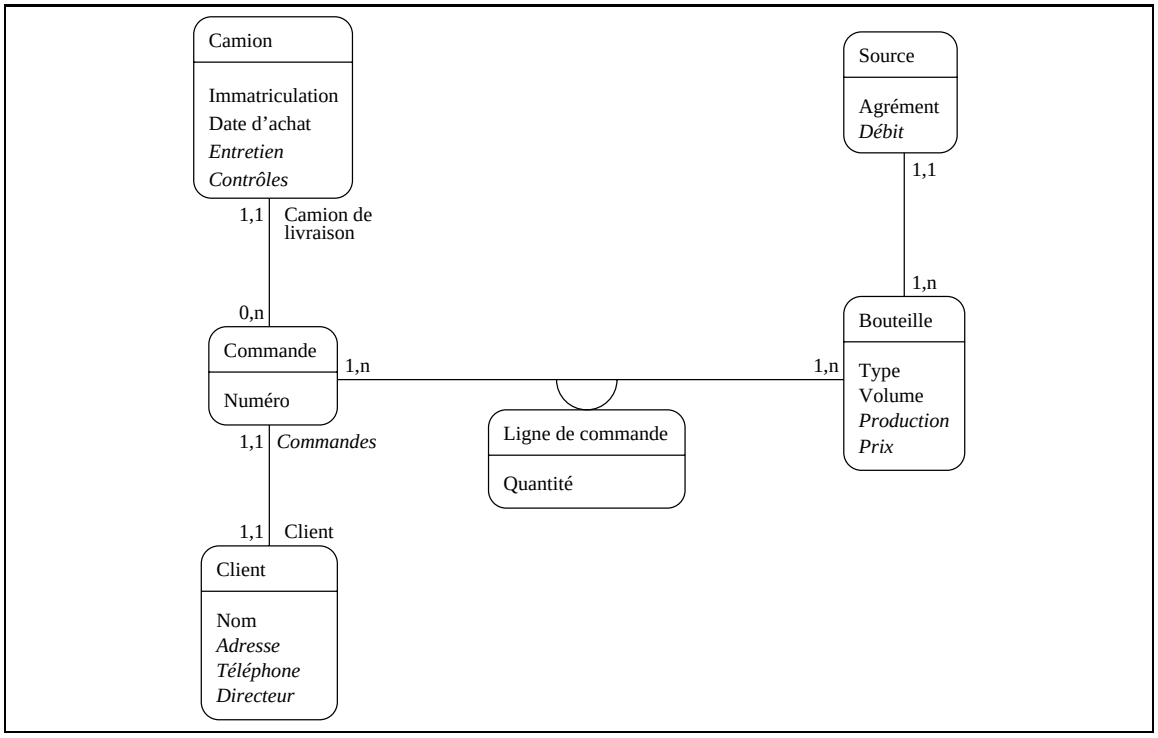


FIG. A.1 – *Modélisation OMT*

Remarque : Dans un contexte non-temporel, les attributs historiques sont gérés explicitement par les applications, en particulier au travers d'attributs multivalués. Ce cas est illustré pour l'historisation des commandes des clients par la figure A.2a.

Dans un modèle offrant la notion d'historique, il ne faut plus utiliser d'attribut multivalué mais simplement un attribut historique (monovalué) comme le montre la figure A.2b. Pour un client donné, le domaine structurel de l'attribut *Commandes* décrit l'ensemble des commandes qu'il a passées. Cet historique étant observé au niveau du jour, il n'est possible d'enregistrer qu'une seule commande par jour.

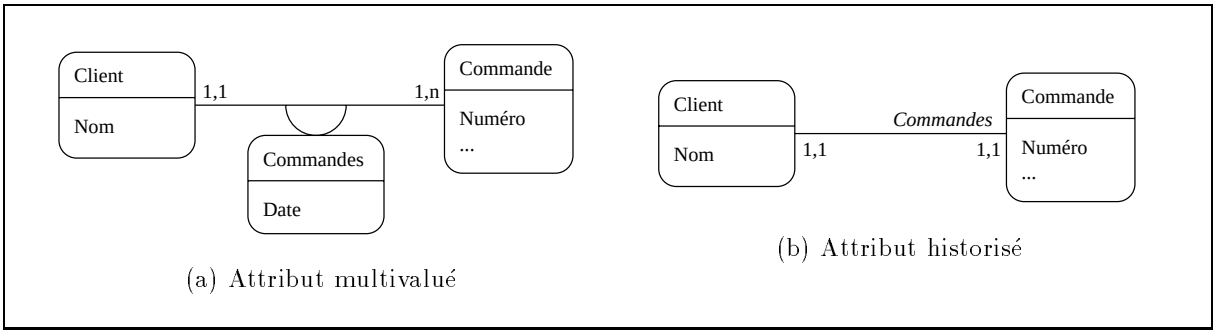


FIG. A.2 – *Attribut multivalué vs attribut historique*

Nous détaillons maintenant les principaux éléments de cette application

A.1.1 La source (Cf. figure A.3)

L'exploitation de la source est soumise à l'obtention d'un agrément dont on ne retient que le numéro.

Afin de déterminer le rythme des chaînes de productionΓle débit de le source est analysé régulièrement. Toutes les douze heuresΓun capteur enregistre le débit horaire d'eau (en m^3). En cas de panneΓil peut y avoir des valeurs absentes. A tout instant (heure)Γon peut demander le débitΓqui est estimé à partir des dernières données enregistrées. Le domaine temporel de cet historique commence le 1^{er} janvier 1990 (date de création de Oplate) et la borne supérieure n'est pas fixée ($+\infty$). Il est donc possible d'interpoler des valeurs pour le futur.

A.1.2 Les bouteilles (Cf. figure A.4)

L'usine produit différents types de bouteillesΓchacun identifié par son nom.

Chaque type est caractérisé par son prix unitaire de vente et sa capacité. L'historique des prix est conservé. La capacité peut changer (par exempleΓlors d'une promotionΓla capacité d'une bouteille "Zébu" peut passer de 1 litre à 1Ł5 litres)Γmais cette évolution n'est pas historisée.

Chaque jourΓl'entreprise enregistre la quantité de bouteilles produites pour chaque type. Les jours où il n'y a pas de production (week-endΓgrêve)Γaucun enregistrement n'est fait dans la base de données.

Le domaine temporel des historiques prix et production commence lors de la création du type de bouteille correspondant et ne sont pas limités dans le futur. La borne supérieure est fixée lorsqu'un type de bouteille n'est plus produit.

A.1.3 Les clients (Cf. figure A.5)

Les sociétés clientes sont identifiées par leur nom. Elles sont enregistrées dans la base lors de leur premier achat à Oplate.

On garde l'historique de leurs adresses et de leurs numéros de téléphone. Ces deux attributs évoluent de manière synchrone.

On connaît les directeurs successifs des entreprises clientes (ce renseignement est historisé).

On enregistre également l'historique des ventes qui leur sont faites. Cet historique est observé au niveau du jourΓce qui signifie que les commandes passées un même jour par un client donné sont regroupées.

Le domaine temporel de tous les attributs relatifs aux clients commence lors de la première commande du client et n'est pas borné dans le futur.

A.1.4 Les commandes (Cf. figure A.6)

Les commandes sont identifiées par un numéro.

Lorsqu'il passe une commandeΓun client peut acheter des bouteilles de divers types. Pour chaque typeΓil indique la quantité de bouteilles voulue. Le montant de la commande est calculé automatiquement en fonction du prix des types commandésΓau jour de la commandeΓet des quantités voulues pour chaque type.

Oplate livre elle-même ses clientsΓet pour chaque commandeΓon enregistre le camion devant effectuer la livraison. On suppose ici qu'une commande peut toujours être livrée par un seul camion.

A.1.5 Les camions (Cf. figure A.7)

Ils sont identifiés par leur numéro d'immatriculation.

On enregistre la date de leur achatΓqui permet de déterminer ensuite la date des contrôles techniques qu'ils doivent subir. La validité du contrôle technique est de 1 an. Les résultats des contrôles (numéro d'agrément obtenu après le contrôle et observations éventuelles) sont historisés au niveau du mois. Il s'agit d'une précision suffisante pour déterminer les prochains contrôles. Ainsi le domaine temporel de cet attribut correspond à une séquence périodique de premier instant la date d'achat (approximée au mois) et de période 1 an.

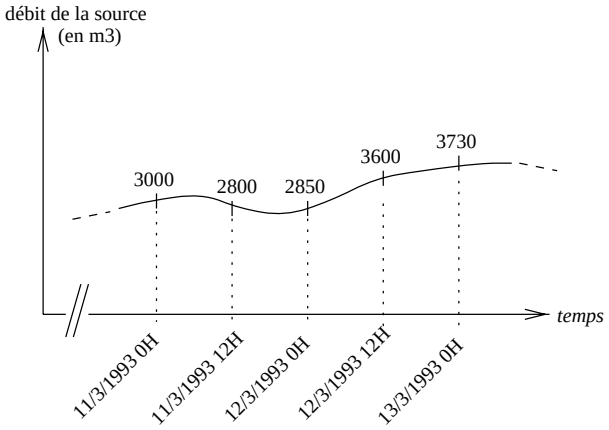
Par ailleursΓles camions sont entretenus par un garagiste. Les dates des révisions ne sont pas connues à l'avanceΓelles dépendent du nombre de livraisons effectuées (kilométrage parcouru) et des incidents éventuels (accidentsΓcasse mécaniqueΓetc). Ces visites sont historisées (nature d'opération et coût). Le domaine temporel des entretiens commencent à la date d'achat du camion. La borne supérieure est fixée lorsque le camion est vendu ou mis à la casse.

Attribut	Type	Nature
Agrément	Entier	Constant
Débit	Réel	Historique

(a) Attributs

Attribut historisé	Unité d'observation	Type historique
Débit de la source	Heure	Interpolé

(b) Détail pour les attributs historiques



(c) Exemple d'évolution du débit de la source :

Attribut	Valeur
Agrément	100450
Débit	[...
	< 11/3/1993 0HΓ3000 >Γ< 11/3/1993 12HΓ2800 >Γ
	< 12/3/1993 0HΓ2850 >Γ< 12/3/1993 12HΓ3600 >Γ
	< 13/3/1993 0HΓ3730 >Γ
	...]

(d) Exemple de valeur

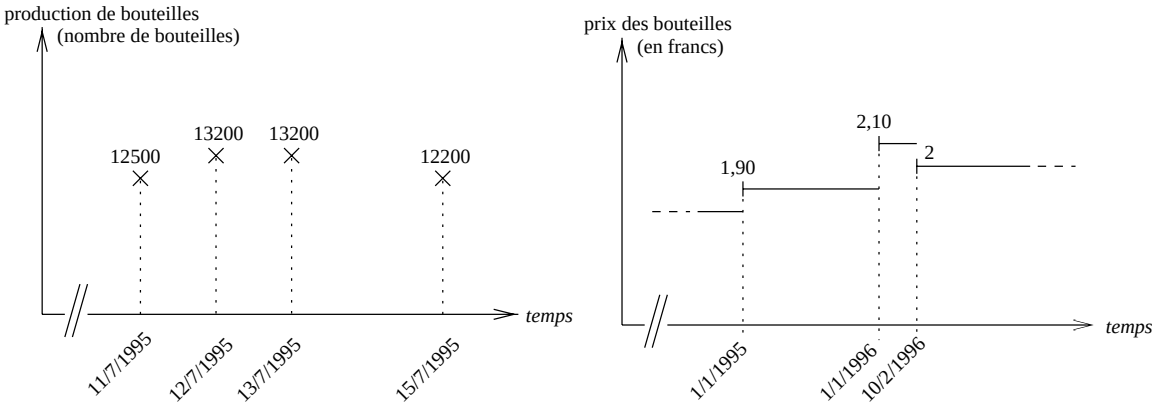
FIG. A.3 – *La source*

Attribut	Type	Nature
Type	Texte	Constant
Volume	Réel	Fugitif
Production	Entier	Historique
Prix	Réel	Historique

(a) Attributs

Attribut historisé	Unité d'observation	Type historique
Production	Jour	Discret
Prix	Jour	Escalier

(b) Détail pour les attributs historiques



(c) Exemple d'évolution du prix et de la production des bouteilles "Plus Soif" :

Attribut	Valeur
Type	PlusSoif
Volume	1l5
Production	[...Γ< 11/7/1995Γ12500 >Γ< 12/7/1995Γ13200 >Γ < 13/7/1995Γ13200 >Γ< 15/7/1995Γ12200 >Γ...]
Prix	[...Γ< 1/1/1995Γ1l90 >Γ< 1/1/1996Γ2l10 >Γ< 10/2/1996Γ2 >]
Type	Zébu
Volume	2l5
Production	[...Γ< 25/10/1996Γ4700 >Γ...]
Prix	[< 1/1/1990Γ3 >Γ< 15/6/1993Γ3l20 >Γ< 1/5/1995Γ3l50 >Γ < 1/5/1997Γ3l80 >]

(d) Exemple de valeur

FIG. A.4 – *Les bouteilles*

Attribut	Type	Nature
Nom	Texte	Constant
Directeur	Texte	Historique
Adresse	Texte	Historique
Téléphone	texte	Historique
Commandes	Commande	Historique

(a) Attributs

Attribut historisé	Unité d'observation	Type historique
Directeur	Jour	Escalier
Adresse	Jour	Escalier
Téléphone	Jour	Escalier
Commandes	Jour	Discret

(b) Détail pour les attributs historiques

Attribut	Valeur
Nom	Supermarché du désert
Directeur	[< 1/1/1990Γ Hiddid >]
Adresse	[< 1/1/1990Γ Tataouine >Γ< 1/12/1994Γ Tunis >]
Téléphone	[< 1/1/1990Γ 162 243 >Γ< 1/12/1994Γ 325 478 >]
Commandes	[< 1/1/1990Γ @00001 >Γ ... < 15/4/1997Γ @24023 >Γ< 10/5/1997Γ @24345 >Γ < 6/6/1997Γ @24589 >]
Nom	Alimentation du maquis
Directeur	[< 1/3/1990Γ Dupond >Γ< 16/8/1992Γ Nil >] < 1/9/1992Γ Maurizi >]
Adresse	[< 1/1/1990Γ Propriano >Γ < 21/5/1992Γ Ajaccio >Γ < 1/4/1993Γ Propriano >]
Téléphone	[< 1/1/1990Γ 04 95 34 12 56 >Γ < 21/5/1992Γ 04 95 56 39 74 >Γ < 1/4/1993Γ 04 95 44 57 22 >]
Commandes	[< 1/3/1990Γ @00054 >Γ< 23/5/1990Γ @00058 >Γ...]

(c) Exemple de valeur

FIG. A.5 – *Les clients*

Attribut	**Type**	**Nature**
Numéro	Entier	Constant
Client	Client	Fugitif
Détail	liste(tuple(	Fugitif
	type : BouteilleΓ	
	quantité : Entier))	
Livraison	Camion	Fugitif

(a) Attributs

Attribut	**Valeur**
Numéro	24023
Client	@Supermarché du désert
Détail	[< 250Γ@PlusSoif >Γ< 100Γ@Zébu >]
Livraison	@4856VG38
Numéro	24589
Client	@Supermarché du désert
Détail	[< 1000Γ@Zébu >]
Livraison	

(b) Exemple de valeur

FIG. A.6 – *Les commandes*

Attribut	**Type**	**Nature**
Immatriculation	Texte	Constant
Date d'achat	Instant(jour)	Fugitif
Entretien	tuple(	Historique
	nature : TexteΓ	
	prix : Réel)	
Contrôle	tuple(	Historique
	agrément : TexteΓ	
	remarques : Texte)	

(a) Attributs

Attribut historisé	**Unité d'observation**	**Type historique**
Entretien	Jour	Discret
Contrôle	Mois	Régulier

(b) Détail pour les attributs historiques

Attribut	**Valeur**
Immatriculation	4856 VG 38
Date d'achat	15/2/1993
Entretien	[...Γ< 5/2/1996Γ< VidangeΓ1200 > >Γ
	< 3/11/1998Γ< BatterieΓ800 > >Γ
	< 8/4/1997Γ< VidangeΓ1300 > >]
Contrôles	[< 8/1995Γ< n48534Γ- > >Γ
	< 8/1996Γ< n67810Γpneux AV usés > >Γ
	< 8/1997Γ< n80143Γ- > >]
Immatriculation	735 AOC 38
Date d'achat	19/5/1997
Entretien	[< 10/07/1997Γ< CarosserieΓ8600 > >]
Contrôles	[ ]

(c) Exemple de valeur

FIG. A.7 – *Les camions*

A.1.6 Quelques requêtes

Nous présentons ici quelques requêtes significatives dans le cadre de cette application :

R.1 : *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

R.2 : *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

R.3 : *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

On considère qu'une bouteille est en vente lorsque son prix de vente est défini.

R.4 : *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

Un client est enregistré dans la base lors de se première commande (il est alors "connu").

R.5 : *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

R.6 : *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

R.7 : *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

R.8 : *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

R.9 : *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

A.2 Modélisation en relationnel (Oracle)

A.2.1 Schéma

Pour transformer le schéma OMT en schéma relationnelΓafin d'éviter toute redondance d'information et pour obtenir un nombre minimal de relationsΓnous avons suivi les règles suivantes (toutes les relations obtenues sont en 3FN) :

- Pour chaque classe de l'applicationΓune relation "historique" (contenant un attribut *validité*) est associée à chaque attribut historique (mono-valué ou multi-valué)Γceci pour éviter les redondances liées aux évolutions asynchrones des attributs.

- Dans chaque relation historiqueΓun nuplet représente un I_instantané <v, t>Γtel que la valeur v est vraie à l'instant t dans le monde réel. Afin de ne pas rendre encore plus complexe le schéma et les requêtesΓnous avons choisi de ne pas représenter les ensembles d'instants (intervalles ou éléments temporels). AinsiΓseuls les changements de valeur sont enregistrés et la valeur d'une relation temporelle entre deux n-uplets est interprétée différemment selon le type (discret ou en escalier) de l'historique modélisé. Pour un historique discretΓil n'y a pas de valeur autre que les valeurs enregistrées. Pour un historique en escalierΓla valeur de la relation entre deux n-uplets consécutifs (suivant leur temps de validité) correspond à la valeur du n-uplet le plus ancien.

- Pour chaque classe de l'applicationΓles attributs mono-valués constants ou fugitifs sont regroupés dans une relation non-historique.

- EnfinΓ une relation non-historique est introduite pour chaque attribut multi-valué constant ou fugitif.

- Lorsqu'une relation non-historique ne comporte qu'un attribut constantΓelle est éliminée car elle est forcément incluse dans une relation "historique".

Nous obtenons ainsi le schéma relationnel suivant :

Source (Validite, Agrement, Debit)

{ $<v, a, d> \in$ *Source* $\iff$ *le débit de la source de numéro d'agrément a, à l'heure v, est d* }

Bouteille (Type, Volume)

{ $<t, v> \in$ *Bouteille* $\iff$ *la bouteille de type t a une capacité de v litres* }

BouteilleProd (Validite, Type, Production)

{ $<v, t, p> \in$ *BouteilleProd* $\iff$ *la bouteille de type t a été produite en p exemplaires le jour v* }

BouteillePrix (Validite, Type, Prix)

{ $<v, t, p> \in$ *BouteillePrix* $\iff$ *le prix unitaire de vente de la bouteille de type t est passé à p francs le jour v* }

Camion (Immat, DateAchat)

{ $<i, d> \in Camion \iff$ *le camion d'immatriculation* i *a été acheté le jour* d }

CamionEntretiens (Validite, Immat, Nature, Prix)

{ $<v, i, n, p> \in CamionEntretiens \iff$ *le camion d'immatriculation* i *a subi une visite d'entretien de type* n *qui a coûté* p *francs le jour* v }

CamionControles (Validite, Immat, Agrement, Remarques)

{ $<v, i, a, r> \in CamionControles \iff$ *le camion d'immatriculation* i *a subi un contrôle technique le mois* v*, l'agrément obtenu est le numéro* a*, et les remarques* r *ont été faites par le contrôleur* }

LivraisonCom (Numero, Client, ImmatCamion)

{ $<n, c, i> \in LivraisonCom \iff$ *la commande de numéro* n *vient du client de nom* c*, et a été livrée par le camion d'immatriculation* i }

LigneCom (NumCom, NumL, TypeB, Quantite)

{ $<c, l, t, q> \in LigneCom \iff$ *la ligne* l *de la commande* c *correspond à* q *exemplaires de bouteilles de type* t }

ClientDirecteur (Validite, Nom, Directeur)

{ $<v, n, d> \in ClientDirecteur \iff$ *la personne* d *a été nommée directeur de la société cliente de nom* n *le jour* v }

ClientCoordonnees (Validite, Nom, Adresse, Telephone)

{ $<v, n, a, t> \in ClientCoordonnes \iff$ *le client de nom* n *a déménagé à l'adresse* a *le jour* v*, son nouveau numéro de téléphone est* t }

ClientCommandes (Validite, Nom, Commandes)

{ $<v, n, c> \in ClientCommandes \iff$ *le client de nom* n *a passé la commande de numéro* c *le jour* v }

La figure A.8 monte un exemple de valeur pour les relations BouteilleProd et BouteillePrix.

BouteilleProd

Validité	Type	Production
...		
11/7/1995	PlusSoif	12500
12/7/1995	PlusSoif	13200
13/7/1995	PlusSoif	13200
15/7/1995	PlusSoif	12200
...		

BouteillePrix

Validité	Type	Prix
1/1/1995	PlusSoif	1[illegible]90
1/1/1996	PlusSoif	2[illegible]10
10/2/1996	PlusSoif	2

FIG. A.8 – *Exemple de valeur pour les relations* BouteilleProd *et* BouteillePrix

La création du schéma en SQL est la suivante :

```
create table Source (
    Validite date,
    Agrement char(20),
    Debit number(4,2),
    constraint Cle_Source
        primary key (Validite, Agrement) ) ;

create table Bouteille (
    Type char(20),
    Volume number(4,2),
    constraint Cle_Bouteille
        primary key (Type) ) ;

create table BouteilleProd (
    Validite date,
    Type char(20)
        references Bouteille(Type),
    Production number(4),
    constraint Cle_BouteilleProd
        primary key (Validite, Type) ) ;

create table BouteillePrix (
    Validite date,
    Type char(20)
        references Bouteille(Type),
    Prix number(4,2),
    constraint Cle_BouteillePrix
        primary key (Validite, Type) ) ;

create table Camion (
    Immat char(20),
    DateAchat date,
    constraint Cle_Camion
        primary key (Immat) ) ;

create table CamionEntretiens (
    Validite date,
    Immat char(20)
        references Camion(Immat),
    Nature char(20),
    Prix number(4,2),
    constraint Cle_CamionEntretiens
        primary key (Validite, Immat) ) ;

create table CamionControles (
    Validite date,
    Immat char(20)
        references Camion(Immat),
    Agrement char(20),
    Remarques char(20),
    constraint Cle_CamionControles
        primary key (Validite, Immat) ) ;

create table LivraisonCom (
    Numero number(4),
    Client char(20),
    ImmatCamion char(20),
    constraint Cle_LivraisonCom
        primary key (Numero) ) ;

create table LigneCom (
    NumCom number(4),
    NumL number(4),
    TypeB char(20),
    Quantite number(4),
    constraint Cle_LigneCom
        primary key (NumCom, NumL) ) ;

create table ClientDirecteurs (
    Validite date,
    Nom char(20),
    Directeur char(20),
    constraint Cle_ClientDirecteurs
        primary key (Validite, Nom) ) ;

create table ClientCoordonnees (
    Validite date,
    Nom char(20)
        references Client(Nom),
    Adresse char(200),
```

```
    Telephone char(20),
    constraint Cle_ClientCoordonnees
        primary key (Validite, Nom) );

create table ClientCommandes (
    Validite date,
    Nom char(20)
        references Client(Nom),
    Commandes number(4),
    constraint Cle_ClientCommandes
        primary key (Validite, Nom) );
```

A.2.2 Requêtes

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

```
SELECT C.Immat, C.DateAchat, E.Nature, E.Prix
FROM Camion C, CamionEntretiens E
WHERE C.Immat = E.immat
```

Cette requête ne contient pas d'information temporelle. Il suffit de mettre en correspondance les informations sur les camions et celles sur leurs visites d'entretien.

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

```
SELECT C.Immat, E.Nature, E.Prix, E.Validite
FROM Camion C, CamionEntretiens E
WHERE C.Immat = E.immat AND E.Nature ≠ 'Vidange'
```

Ici la sélection n'est pas temporelleΓelle s'exprime simplement en SQL. Il suffit de récupérer les instants de validité associés aux n-uplets sélectionnés.

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

```
PrixPeriode =
{ Cette requête permet de construire l'historique des prix des bouteilles dans [1/1/1990,
31/12/1996]. }
SELECT *
{ Ensemble des prix des types de bouteilles modifiés dans la période. }
FROM BouteillePrix
WHERE Validite > '1/1/1990' AND Validite ≤ '1/12/1996'
```

```
UNION
SELECT '1/1/1990', Type, Prix
{ Pour chaque type de bouteille, détermination du prix valide au 1/1/1990 (lorsqu'il est défini
à cette date là). }
FROM BouteillePrix P1
WHERE Validite = (SELECT MAX(Validite)
                  FROM BouteillePrix P2
                  WHERE Validite ≤ '1/1/1990'
                  AND P2.Type = P1.Type)
UNION
SELECT '1/1/1990', Type, NULL
{ Pour chaque type de bouteille dont le prix n'est pas défini au 1/1/1990, construction d'un
n-uplet avec la valeur absente. }
FROM BouteillePrix P1
WHERE Type NOT IN (SELECT Type
                   FROM BouteillePrix
                   WHERE Validite < '1/1/1990'
```

Requête finale :

```
SELECT *
FROM PrixPeriode
WHERE Type NOT IN (SELECT Type
                   FROM PrixPeriode
                   WHERE Prix IS NULL)
```

La requête finale retient le type et l'historique des prix pour les bouteilles dont le prix n'a jamais été absent (donc jamais à NULL) pendant la période donnée.

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

```
SELECT D.Nom, D.Directeur, D.Validité
FROM ClientDirecteur D, ClientCoordonnees C
WHERE C.Nom = D.Nom AND C.adresse = 'Paris'
AND D.Nom in (SELECT Nom
              FROM ClientCommandes
              WHERE Validite ≤ '31/12/1994')
{ La condition précédente permet de ne retenir que les clients connus depuis 1994. }
AND (C.Validite = D.Validite
     { La condition précédente permet de sélectionner les n-uplets correspondant aux clients
     ayant simultanément changé de directeur et d'adresse. }
     OR
```

```
C.Validite < D.Validite
AND NOT EXISTS (SELECT *
                FROM ClientCoordonnees CC
                WHERE CC.Nom = C.Nom
                AND CC.Validite > C.Validite
                AND CC.Validite ≤ D.Validite)
```
{ *La condition précédente permet de sélectionner les n-uplets correspondant aux clients dont le directeur a changé lorsqu'ils étaient déjà à Paris.* }
```
OR
D.Validite < C.Validite
AND NOT EXISTS (SELECT *
                FROM ClientDirecteur DD
                WHERE DD.Nom = D.Nom
                AND DD.Validite > D.Validite
                AND DD.Validite ≤ C.Validite))
```
{ *La condition précédente permet de sélectionner les n-uplets correspondant aux clients dont le directeur est en fonction au moment d'un déménagement à Paris.* }

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

```
SELECT P.Validite
FROM BouteilleProd P, BouteilleProd Z
WHERE P.Type = 'PlusSoif' AND Z.Type = 'Zébu'
AND P.Validite (+) = Z.Validite
AND P.Production > NVL(Z.Production, 0)
```

On utilise ici le produit externe (clause (+) dans l'expression du produit) pour sélectionner les instants pendant lesquels les bouteilles "PlusSoif" étaient produites alors que les bouteilles "Zébu" ne l'étaient pas.

Le résultat de cette requête est un ensemble d'instants. SQL ne dispose pas d'un type élément temporel permettant de les regrouper en une seule valeur.

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

```
SELECT O.Validité, O.Type, O.Production, I.Prix
FROM BouteilleProd O, BouteillePrix I
WHERE O.Type = I.Type
AND O.Validité ≥ '1/1/97' AND O.Validité ≤ '31/12/97'
```
{ *La disjonction ci-dessous permet de déterminer le prix d'une bouteille à chaque instant où une valeur de sa production est donnée. Les n-uplets retenus sont ceux pour lesquels le prix correspond à l'instant de la production, soit t_1, que le prix ait été changé à cet instant où à un instant précédent, soit t_2 (sans autre modification de prix entre t_1 et t_2).* }

```
AND (I.Validité = O.Validité
        OR
        I.Validité < O.Validité
        AND NOT EXISTS (SELECT *
                        FROM BouteillePrix II
                        WHERE II.Type = I.Type
                        AND II.Validité > I.Validité
                        AND II.Validité ≤ O.Validité))
```

La requête manipule deux informations qui n'évoluent pas de manière synchrone. Pour les productions enregistrées pendant la période considéréeΓla requête calcule à chaque fois le prix correspondant.

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

```
PrixProduction =
SELECT O.Validité, O.Type, O.Production, I.Prix
FROM BouteilleProd O, BouteillePrix I
WHERE O.Type = I.Type
AND (I.Validité = O.Validité
        OR
        I.Validité < O.Validité
        AND NOT EXISTS (SELECT *
                        FROM BouteillePrix II
                        WHERE II.Type = I.Type
                        AND II.Validité > I.Validité
                        AND II.Validité ≤ O.Validité))
```

{ *Cette requête intermédiaire, selon le même principe que R.6 réunit les historiques de production et de prix (sans restreindre à un domaine temporel particulier).* }

```
Requête finale :
SELECT SUM(Production)
FROM PrixProduction
WHERE Type = 'PlusSoif' AND Prix > 5
```

Cette requête restreint la relation au type et au prix considéréΓet calcule la somme des bouteilles produites.

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

```
SELECT Nature, Validité
FROM CamionEntretiens
WHERE E1.Immat = '735 AOC 38'
AND Validité > (SELECT MIN(Validité)
                FROM CamionEntretiens
                WHERE Immat = '735 AOC 38'
                AND Nature = 'vidange')
```

Cette requête sélectionne les visites effectuées sur le camion concerné après la date de la première vidange (calculée par la sous-requête).

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

```
SELECT C2.Adresse
FROM ClientCoordonnees C1, ClientCoordonnees C2
WHERE C1.Nom = C2.Nom
AND C1.Nom = 'Alimentation du maquis'
AND C1.Adresse = 'Propriano'
AND C1.Validite < C2.Validite
AND NOT EXISTS (SELECT *
                FROM ClientCoordonnees C3
                WHERE C3.Nom = C1.Nom
                AND C3.Validite > C1.Validite
                AND C3.Validite < C2.Validite)
```

Le requête principale fait le lien entre l'adresse à Propriano et les adresses suivantes. La sous-requête permet de sélectionner le n-uplet C2 correspondant à l'adresse où le client a déménagé juste après avoir quitté Propriano.

A.3 Modélisation en TSQL 2

A.3.1 Schéma

Le fait d'associer le temps de validité aux n-uplets dans TSQL 2 nous contraint à créer plusieurs tables lorsque dans une relation l'évolution des attributs historiques n'est pas synchroneΓet en particulier lorsqu'ils ne sont pas observés à la même granularité.

De ce faitΓle schéma relationnel obtenu est identique à celui de SQL 2Γà ceci près que la gestion du temps de validité est pris en charge par le système.

Suivant le type d'historique voluΓTSQL 2 offre deux types de relations : les *valid event relations*Γqui correspondent aux historiques discretsΓet les *valid state relations*Γqui correspondent aux historiques en escalier. Pour les historiques interpolésΓil faut stocker les valeurs effectives comme pour un historique discret et faire ensuite l'interpolation lors de l'interrogation.

Le schéma relationnel final est le suivant :

Source (Agrement, Debit) { *valid event table* }
{ <*a, d, v*> ∈ *Source* ⟺ *le débit de la source de numéro d'agrément a est d, à tous les instant de v* }
{ *L'attribut relatif au temps de validité est automatiquement ajouté par TSQL 2 pour les relations historiques. Il s'agit d'un élément temporel* }

Bouteille (Type, Volume) ¿{ *snapshot table* }
{ <*t, v*> ∈ *Bouteille* ⟺ *la bouteille de type t a une capacité de v litres* }

BouteilleProd (Type, Production) { *valid event table* }
{ <*t, p, v*> ∈ *BouteilleProd* ⟺ *la bouteille de type t a été produite en p exemplaires chaque jour de v* }

BouteillePrix (Type, Prix) { *valid state table* }
{ <*t, p, v*> ∈ *BouteillePrix* ⟺ *le prix unitaire de vente de la bouteille de type t est de à p francs pendant la période v* }

Camion (Immat, DateAchat) { *snapshot table* }
{ <*i, d*> ∈ *Camion* ⟺ *le camion d'immatriculation i a été acheté le jour d* }

CamionEntretiens (Immat, Nature, Prix) { *valid event table* }
{ <*i, n, p, v*> ∈ *CamionEntretiens* ⟺ *le camion d'immatriculation i a subi une visite d'entretien de type n qui a coûté p francs aux instants de v* }

CamionControles (Immat, Agrement, Remarques) { *valid event table* }
{ <*i, a, r, v*> ∈ *CamionControles* ⟺ *le camion d'immatriculation i a subi un contrôle technique aux instants de v, l'agrément obtenu est le numéro a, et les remarques r ont été faites par le contrôleur* }

LivraisonCom (Numero, Client, ImmatCamion) { *snapshot table* }
{ <*n, c, i*> ∈ *LivraisonCom* ⟺ *la commande de numéro n vient du client de nom c, et a été livrée par le camion d'immatriculation i* }

LigneCom (NumCom, NumL, TypeB, Quantite) { *snapshot table* }

{ <*c, l, t, q*> ∈ *LigneCom* ⟺ *la ligne l de la commande c correspond à q exemplaires de bouteilles de type t* }

ClientDirecteur (Nom, Directeur) { *valid state table* }

{ <*n, d, v*> ∈ *ClientDirecteur* ⟺ *la personne d est directeur de la société cliente de nom n pendant la période v* }

ClientCoordonnes (Nom, Adresse, Telephone) { *valid state table* }

{ <*n, a, t, v*> ∈ *ClientCoordonnes* ⟺ *a est l'adresse et t le téléphone du client de nom n pendant la période v* }

ClientCommandes (Nom, Commandes) { *valid event table* }

{ <*n, c, v*> ∈ *ClientCommandes* ⟺ *le client de nom n a passé la commande de numéro c le jour v* }

La figure A.9 monte un exemple de valeur pour les relations BouteilleProd et BouteillePrix.

BouteilleProd

Type	**Production**	*Validité*
...		
PlusSoif	12500	11/7/1995
PlusSoif	13200	12/7/1995
PlusSoif	13200	13/7/1995
PlusSoif	12200	15/7/1995
...		

BouteillePrix

Type	**Prix**	*Validité*
PlusSoif	1Ł90	[1/1/1995Γ31/12/1995]
PlusSoif	2Ł10	[1/1/1996Γ9/2/1996]
PlusSoif	2	[10/2/1996Γnow]

FIG. A.9 – *Exemple de valeur pour les relations* BouteilleProd *et* BouteillePrix

La définition du schéma en TSQL 2 est :

```
create table Source (
    Agrement char(20),
    Debit number(4,2),
    constraint Cle_Source
        primary key (Agrement) )
as valid event to hour ;

create table Bouteille (
    Type char(20),
    Volume number(4,2),
    constraint Cle_Bouteille
        primary key (Type) ) ;
```

```
create table BouteilleProd (
    Type char(20)
        references Bouteille(Type),
    Production number(4),
    constraint Cle_BouteilleProd
        primary key (Type) )
as valid event to day ;

create table BouteillePrix (
    Type char(20)
        references Bouteille(Type),
    Prix number(4,2),
```

```
    constraint Cle_BouteillePrix
        primary key (Type) )
 as valid state to day ;

create table Camion (
    Immat char(20),
    DateAchat date,
    constraint Cle_Camion
        primary key (Immat) ) ;

create table CamionEntretiens (
    Immat char(20)
        references Camion(Immat),
    Nature char(20),
    Prix number(4,2),
    constraint Cle_CamionEntretiens
        primary key (Immat) )
as valid event to day ;

create table CamionControles (
    Immat char(20)
        references Camion(Immat),
    Agrement char(20),
    Remarques char(20),
    constraint Cle_CamionControles
        primary key (Immat) )
as valid event to month ;

create table LivraisonCom (
    Numero number(4),
    Client char(20),
    ImmatCamion char(20),
    constraint Cle_Commande
        primary key (Numero) ) ;

create table LigneCom (
    NumCom number(4),
    NumL number(4),
    TypeB char(20),
    Quantite number(4),
    constraint Cle_LigneCom
        primary key (NumCom, NumL) ) ;

create table ClientDirecteurs (
    Nom char(20),
    Directeur char(20),
    constraint Cle_ClientDirecteurs
        primary key (Nom) )
as valid state to day ;

create table ClientCoordonnées (
    Nom char(20)
        references ClientDirecteurs(Nom),
    Adresse char(200),
    Telephone char(20),
    constraint Cle_ClientCoordonnées
        primary key (Nom) )
as valid state to day ;

create table ClientCommandes (
    Nom char(20)
        references ClientDirecteurs(Nom),
    Commandes number(4),
    constraint Cle_ClientCommandes
        primary key (Nom) )
as valid event to day ;
```

A.3.2 Requêtes

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

```
SELECT C.Immat, C.DateAchat, E.Nature, E.Prix
FROM Camion C, CamionEntretiens E
WHERE C.Immat = E.Immat
```

Le produit d'une relation de type *event table* avec une autre relation est une relation de type *event table*.

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

```
SELECT C.Immat, E.Nature, E.Cout
FROM Camions C, CamionEntretiens E
WHERE C.Immat = E.Immat
AND E.Nature != 'vidange'
```

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

```
SELECT B.Type, B.Prix VALID INTERSECT (PERIOD '[1/1/1990, 1/12/1996]')
{ La clause VALID INTERSECT effectue la restriction temporelle à la période donnée de la
relation résultat de la requête. }
FROM BouteillePrix B
WHERE VALID(B) CONTAINS PERIOD '[1/1/1990, 1/12/1996]'
{ La clause WHERE introduit ici une sélection temporelle. }
```

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

```
SELECT D.Nom, D.Directeur
FROM ClientDirecteurs D, ClientCoordonnées C
WHERE D.Nom = C.Nom
AND D.Nom IN (SELECT CC.Nom
              FROM ClientCommandes CC
              WHERE valid(CC) < '1/1/1994'
```

L'emboitement de la sous-requête est nécessaire du fait de l'extension du produit cartésion en TSQL 2 (sémantique de l'intersection). La requête ci-dessous construit une relation de type *event table* et restreint l'historique résultat aux instants auxquels est associée une commande :

```
SELECT D.Nom, D.Directeur
FROM ClientDirecteurs D, ClientCoordonnées C, ClientCommandes Com
WHERE D.Nom = C.Nom AND D.Nom = Com.Nom
AND VALID(Com) < '1/1/1994'
```

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

```
SELECT VALID(P1)
FROM BouteilleProd P1, BouteilleProd P2
WHERE P1.Type = 'PlusSoif' AND P2.Type = 'Zébu'
AND P1.Production > P2.Production
UNION
SELECT VALID(P1)
FROM BouteilleProd P1
WHERE P1.Type = 'PlusSoif'
AND NOT EXISTS (SELECT *
                FROM BouteilleProd P2
                WHERE P2.Type = 'Zébu'
                AND VALID(P1) = VALID(P2))
```

La seconde partie de la requête permet de sélectionner les instants où les bouteilles "PlusSoif" sont produites alors que les bouteilles "Zébu" ne le sont pas.

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

```
SELECT P.Type, O.Production, P.Prix VALID INTERSECT '[1/1/1997'..'31/12/1997]'
FROM BouteillePrix P, BouteilleProduction O
WHERE P.Type = O.Type
```

TSQL 2 est particulièrement bien adapté à l'expression de requêtes dans lesquelles on combine des instantanés définis aux mêmes instants et issus d'historiques différents.

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

```
SELECT SUM(O.Production)
FROM BouteillePrix P, BouteilleProduction O
WHERE O.Type = P.Type AND P.Prix > 5
```

TSQL 2 est particulièrement bien adapté à l'expression de requêtes dans lesquelles on combine des instantanés issus d'historiques différents et définis aux mêmes instants.

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

```
SELECT E1.Nature
FROM CamionEntretiens E1
WHERE E1.Immat = '735 AOC 38'
AND VALID(E1) > (SELECT MIN(VALID(E2))
                 FROM CamionEntretiens E2
                 WHERE E2.Immat = E1.Immat
                 AND E2.Nature = 'vidange')
```

Ici aussi l'emboitement des deux requêtes est rendu obligatoire par la sémantique du produit cartésien.

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

```
SELECT SNAPSHOT C1.Adresse
FROM ClientCoordonnées C1
WHERE C1.Nom = 'Alimentation du maquis'
AND EXISTS (SELECT *
            FROM C2 in ClientCoordonnées
            WHERE C2.Nom = C1.Nom
            AND C2.Adresse = 'Propriano'
            AND VALID(C1) MEETS VALID(C2))
```

La clause SELECT SNAPSHOT exprime la projection de la relation résultat selon les valeurs structurelles. Le résultat est une relation instantanée.

A.4 Modélisation en TempSQL

A.4.1 Schéma

TempSQL propose deux types de relations :

- les relations instantanées (*snapshot relation*)Γqui correspondent aux relations non temporelles classiques de SQLΓ
- les relations temporelles (*temporal relation*).

Dans les relations temporellesΓTempSQL associe le temps de validité aux attributs. Cela simplifie sensiblement le schéma relationnel obtenuΓcar il n'est plus nécessaire de multiplier les relations comme nous avons dû le faire pour SQL 2 et TSQL 2.

La manipulation d'expressions temporelles en TempSQL est basée sur la manipulation d'éléments temporels. L'opérateur [[E]] définit la période de validité de l'expression E.

Le schéma relationnel TempSQL est le suivant :

Source (Agrement, Debit)
{ *temporal relation* }
{ *Soit* $<a, d> \in$ *Source :*
$<t, v> \in a \iff$ *la source a pour numéro d'agrément v aux instants de t,*
$<t, v> \in d \iff$ *le débit de la source a est v aux instants de t,*
et $[\![a]\!] = [\![d]\!]$ }

Bouteille (Type, Volume, Production, Prix)
{ *temporal relation* }
{ *Soit* $<y, l, o, i> \in$ *Bouteille :*
$<t, v> \in y \iff$ *la bouteille a pour type v aux instants de t,*
$<t, v> \in l \iff$ *le volume de la bouteille y est v aux instants de t,*
$<t, v> \in o \iff$ *la production de la bouteille y est v aux instants de t,*
$<t, v> \in i \iff$ *le prix de la bouteille y est v aux instants de t,*
et $[\![y]\!] = [\![l]\!] = [\![o]\!] = [\![i]\!]$ }

Camion (Immat, DateAchat, Ent_Nature, Ent_Prix, Cont_Agrement, Cont_Rem)
{ *temporal relation* }
{ *Soit* $<i, d, n, p, a, r> \in$ *Camion :*
$<t, v> \in i \iff$ *le camion a pour immatriculation v aux instants de t,*
$<t, v> \in d \iff$ *la date d'achat du camion i est v aux instants de t,*
$<t, v> \in n \iff$ *la nature de l'entretien fait sur le camion i est v aux instants de t,*
$<t, v> \in p \iff$ *le prix de l'entretien fait sur le camion i est v aux instants de t,*
$<t, v> \in a \iff$ *l'agrément du contrôle fait sur le camion i est v aux instants de t,*
$<t, v> \in r \iff$ *les remarques du contrôle fait sur le camion i sont v aux instants de t,*
et $[\![i]\!] = [\![d]\!] = [\![n]\!] = [\![p]\!] = [\![a]\!] = [\![r]\!]$ }

LigneCom (NumCom, NumL, TypeB, Quantite)

{ *snapshot relation* }

{ <*c, l, t, q*> $\in$ *LigneCom* $\Longleftrightarrow$ *la ligne l de la commande c correspond à q exemplaires de bouteilles de type t* }

LivraisonCom (Numero, Client, ImmatCamion)

{ *snapshot relation* }

{ <*n, c, i*> $\in$ *LivraisonCom* $\Longleftrightarrow$ *la commande de numéro n vient du client de nom c, et a été livrée par le camion d'immatriculation i* }

Client (Nom, Directeur, Adresse, Telephone, Commandes)

{ *temporal relation* }

{ *Soit* <*n, d, a, p, c*> $\in$ *Client:*

<*t, v*> $\in n \Longleftrightarrow$ *le client a pour nom v aux instants de t,*

<*t, v*> $\in d \Longleftrightarrow$ *le directeur du client n est v aux instants de t,*

<*t, v*> $\in a \Longleftrightarrow$ *l'adresse du client n est v aux instants de t,*

<*t, v*> $\in p \Longleftrightarrow$ *le numéro de téléphone du client n est v est v aux instants de t,*

<*t, v*> $\in c \Longleftrightarrow$ *la commande du client n est celle de numéro v aux instants de t,*

et $[\![n]\!] = [\![d]\!] = [\![a]\!] = [\![p]\!] = [\![c]\!]$ }

La figure A.10 monte un exemple de valeur pour la relation Bouteille.

TempSQL impose une homogénéïté temporelle des différents attributs. Cela contraint ici à définir une valeur de production et une valeur de prix à chaque instant de la période de validité de la bouteille.

Type	**Volume**	**Production**	**Prix**
[1/1/1995Γnow]: PlusSoif	[1/1/1995Γnow]: 1Π5	[1/1/1995] ∪ [1/5/1995] ∪ [14/7/1995] ∪ [15/8/1995] ∪ ...: 0 [11/7/1995]: 12500 [12/7/1995]: 13200 [13/7/1995]: 13200 [15/7/1995]: 12200 ...	[1/1/1995Γ31/12/1995]: 1Π90 [1/1/1996Γ9/2/1996]: 2Π10 [10/2/1996Γnow]: 2

FIG. A.10 – *Exemple de valeur pour la relation* Bouteille

Nous ne donnons pas la définition du schéma en TempSQL car la syntaxe n'est pas fournie dans [GN93]

A.4.2 Requêtes

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

SELECT Immat, DateAchat, Ent_Nature, Ent_Prix
FROM Camion

Il n'est pas possible d'obtenir un TempSQL une projection sur les valeurs structurelles de la relation. Le résultat de cette requête est une relation temporelle.

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

SELECT Immat, Ent_Nature, Ent_Prix
WHILE [[Ent_Nature $\neq$ 'vidange']]
{ *La clause WHILE introduit la restriction temporelle, à la période donnée, de la relation résultat.* }
FROM Camion

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

SELECT Type, Prix
WHILE [1/1/1990, 1/12/1996]
FROM Bouteille
WHERE [[Prix]] $\subseteq$ [1/1/1990, 1/12/1996]
{ *L'expression [[Prix]] constuit le domaine temporel de l'attribut Prix du n-uplet traité.* }
{ *La condition permet de ne retenir que les n-uplets dont le prix est défini pendant tout l'intervalle.* }

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

SELECT Directeur
WHILE [[Adresse= 'Paris']]
FROM Client
WHERE firstInstant ([[Commandes]]) $\leq$ '31/12/1994'
{ *La fonction firstInstant correspond au premier instant d'un élément temporel.* }

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

```
[[SELECT P.Production
 FROM Bouteille P, Bouteille Z
 WHERE P.Type = 'PlusSoif' and Z.Type = 'Zébu'
 AND P.Production > Z.Production ]]
```

Le jeu d'opérateurs proposé dans [GN93] ne permet pas de prendre en compte les instants où les bouteilles de type "PlusSoif" sont produites alors que celles de type "Zébu" ne le sont pas.

Dans TempSQL la sémantique du produit cartésien est celle de l'intersection (comme TSQL 2).

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

```
SELECT Type, Production, Prix
WHILE [1/1/1997, 31/12/1997]
FROM Bouteille
```

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

```
{ type du résultat : historique de productions }
SELECT Production
WHILE [[Prix > 5 ]]
FROM Bouteille
```

Pour donner la quantité globale produite du type de ces bouteilles il est nécessaire de disposer d'un opérateur de projection d'un historique selon ses valeurs structurelles afin d'appliquer dessus des opérations arithmétiques. Un tel opérateur n'est pas défini dans TempSQL.

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

Les requêtes R.8 et R.9 utilisant un raisonnement basé sur la succession dans le temps ne peuvent pas être exprimées avec les opérateurs définis dans [GN93].

A.5 Modélisation en objets (O_2)

A.5.1 Schéma

Le schéma O_2 est obtenu simplement en traduisant les classes de la modélisation OMT en classe O_2.

La gestion des historiques doit être faite dans la modélisationΓceux-ci sont représentés à l'aide de séquences de couples <valeur, validite>. Dans le cas d'un historique discretΓla valeur de l'attribut est définies qu'aux instants datant les instantanés de son historique. Pour les historiques en escalierΓla valeur entre deux instantanés successifs (selon le temps de validité) est définie comme la valeur de l'instanané le plus ancien des deux.

Le fait de ne disposer en OQL que d'un seul type date à la granularité du jour ne permet pas de faire la différence entre les historiques observés à la granularité du jour et ceux observés à la granularité du mois.

Pour chaque classeΓnous avons défini une racine de persistance permettant de stockerΓ puis d'accéder aux données.

Le schéma O_2 est le suivant :

```
class Source inherit Object
public type tuple (
    Agrement : string,
    Debit : list(tuple(
            Validite: date,
            Valeur: real)))
end ;

class Bouteille inherit Object
public type tuple (
    Type : string,
    Volume : real,
    Production : list(tuple(
            Validite: date,
            Valeur: integer)),
    Prix : list(tuple(
            Validite: date,
            Valeur: real)))
end ;

class Camion inherit Object
public type tuple (
    Immat : string,
    DateAchat : date,
    Entretien : list(tuple(
            Validite: date,
            Valeur: tuple(
                Nature: string,
                Prix: real))),
    Controle : list(tuple(
            Validite: date,
            Valeur: tuple(
                Agrement: string,
                Remarques: string))))
end ;

class Commande inherit Object
public type tuple (
    Numero : integer,
    Client : Client,
    Détail : list (tuple (
            TypeB : Bouteille,
            Quantite : integer)),
    Livraison : Camion)
end ;

class Client inherit Object
public type tuple (
```

```
    Nom : string,
    Directeur : list(tuple(
            Validite: date,
            Valeur: string)),
    Coordonnées : list(tuple(
            Validite: date,
            Valeur: tuple(
                Adresse: string,
                Téléphone: string))),
    Commandes : list(tuple(
            Validite: date,
            Valeur: Commande)))
end ;

name LaSource : Source ;
name LesBouteilles : set (Bouteille) ;
name LesCamions : set (Camion) ;
name LesClients : set (Client) ;
name LesCommandes : set (Commande) ;
```

A.5.2 Requêtes

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

```
{ type du résultat : { < string, date, [<string, real>] > } }
SELECT TUPLE (immat : C.Immat,
              achat: C.DateAchat,
              entretiens: (SELECT TUPLE(nature: E.Valeur.Nature,
                                        prix: E.Valeur.Prix)
                           FROM E IN C.Entretien))
FROM C IN LesCamions
```

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

```
{ type du résultat : { < string, [<date, string, real>] > } }
SELECT TUPLE(Immat : C.Immat,
             Entretiens : SELECT TUPLE(Nature : E.Nature,
                                       Prix : E.Prix,
                                       Validité : E.Validité)
                          FROM E IN C.Entretiens
                          WHERE E.Valeur.Nature ≠ 'vidange')
FROM C IN LesCamions
```

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

```
DEFINE PrixPériode AS
{ Cette requête permet de construire l'historique des prix des bouteilles dans [1/1/1990,
31/12/1996].
Le type du résultat est { < string, [<date, real>] > }. }
```

```
SELECT TUPLE(Type : B.Type,
             HistPrix: SELECT P1 FROM P1 IN B.Prix
                       { Ensemble des prix de bouteille modifiés dans la période. }
                       WHERE P1.Validite ≥ '1/1/1990'
                       AND P1.Validite ≤ '1/12/1996'
                       UNION
                       SELECT TUPLE(Validite: '1/1/1990',
                                    Prix: P2.Prix)
                       { Pour chaque type de bouteille, détermination du prix valide au
                       1/1/1990 (lorsqu'il est défini à cette date là). }
                       FROM P2 in B.Prix
                       WHERE P2.Validite = MAX(SELECT P3.Validite
                                               FROM P3 in B.Prix
                                               WHERE P3.Validite < '1/1/1990')
             )
FROM B IN LesBouteilles
```

Requête finale :

```
{ type du résultat : { < string, [<date, real>] > } }
SELECT H
FROM H IN PrixPériode
WHERE MIN(SELECT P.Validite FROM P IN H.HistPrix) = '1/1/1990'
AND FOR ALL H IN PrixPériode : H.Valeur ≠ nil
```

La requête finale permet de ne retenir que les types de bouteilles dont le prix est défini dès le premier instant de la périodeΓet pendant toute la période.

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

```
{ type du résultat : { < string, [<date, string>] > } }
SELECT TUPLE(Nom : C.Client,
             Directeur : SELECT D
                         FROM D IN C.Directeur
                         WHERE EXISTS A1 in C.Coordonnées :
                                 (A1.Valeur.Adresse = 'Paris'
                                 { La disjonction ci-dessous permet de sélectionner les
                                 directeurs nommés lors d'un déménagement à Paris, ou
                                 nommés lorsque le client était déjà à Paris, ou enfin déjà
                                 en poste lors d'une déménagement du client à Paris. }
                                 AND (A1.Validité = D.Validité
                                      OR
```

```
                                    A1.Validite < D.Validité
                                    AND NOT EXISTS A2 IN C.Coordonnées:
                                    (A2.Validité < D.Validité
                                    AND A2.Validité > A1.Validité)
                                    OR
                                    A1.Validite > D.Validité
                                    AND NOT EXISTS A2 IN C.Coordonnées:
                                    (A2.Validité < A1.Validite
                                    AND A2.Validité > D.Validité)))
            )
FROM C IN LesClients
WHERE EXISTS Com IN C.Commandes: Com.Validité < '31/12/1994'
```

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

```
{ type du résultat: { date } }
SELECT PPS.Validite
FROM PS IN LesBouteilles, Z IN LesBouteilles,
     PPS IN PS.Production, PZ IN Z.Production
WHERE PS.Type = 'PlusSoif' AND Z.Type = 'Zébu'
AND PPS.Validite = PZ.Validite
AND PPS.Valeur > PZ.Valeur
```

{ La permière partie de la requête permet de retenir les instants pendant lesquels la production des bouteilles de type "PlusSoif" est supérieure à celle des bouteilles de type "Zébu". }

```
UNION
SELECT PPS.Validite
FROM PS IN LesBouteilles, PPS IN PS.Production
WHERE PS.Type = 'PlusSoif'
AND PPS.Validite NOT IN (SELECT PZ.Validite
     FROM Z IN LesBouteilles, PZ IN Z.Production
     WHERE Z.Type = 'Zébu')
```

{ La seconde partie de la requête permet de retenir les instants pendant lesquels le type de bouteilles "PlusSoif" est produit alors que le type "Zébu" ne l'est pas. }

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

```
{ type du résultat : { < string, [<date, integer>], [<date, real>] > } }
SELECT TUPLE(Type : B.Type,
             Production : SELECT P
                          FROM P IN B.Production
                          WHERE P.Validite ≥ '1/1/1997' AND P.Validite ≤ '31/12/1997',
                          { Historique de la production pendant l'intervalle donné. }
             Prix : SELECT P
                    FROM P IN B.Prix
                    WHERE P.Validite > '1/1/1997' AND P.Validite ≤ '31/12/1997'
                    UNION
                    SELECT TUPLE (Validite: '1/1/1997',
                                  Valeur: P.Valeur)
                    FROM P IN B.Prix
                    WHERE P.Validite = MAX (SELECT P.Validite
                                            FROM P IN B.Prix
                                            WHERE P.Validite ≤ '1/1/1997'))
                    { Historique des prix pendant l'intervalle donné. Le prix au '1/1/1997'
                    est calculé explicitement car il peut être défini à une date antérieure.
                    }
FROM B IN LesBouteilles
```

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

```
{ type du résultat : integer }
SUM(SELECT Prod.Valeur
    FROM B IN LesBouteilles, Prod IN B.Production, Prx in B.Prix
    WHERE B.Type = 'PlusSoif'
    AND Prx.Valeur > 5
    { La disjonction ci-dessous permet de déterminer le prix d'une bouteille à chaque instant
    où une valeur de sa production est donnée. Les n-uplets retenus sont ceux pour lesquels
    le prix correspond à l'instant de la production, soit t1, que le prix ait été changé à cet
    instant où à un instant précédent, soit t2 (sans autre modification de prix entre t1 et
    t2). }
    AND (Prx.Validite = Prod.Validite
         OR
         Prx.Validite < Prod.Validite
         AND NOT EXISTS Prx2 IN B.Prix :
                        Prx2.Validité > Prx.Validité
                        AND Prx2.Validité < Prod.Validité))
```

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

```
{ type du résultat : [ string ] }
SELECT E1.Valeur
FROM C IN LesCamions, E1 IN C.Entretien
WHERE C.Immat = '735 AOC 38'
AND E1.Validité > MIN(SELECT E2.Validité
                      FROM E2 IN C.Entretien
                      WHERE E2.Valeur.Nature = 'vidange')
```

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

```
{ type du résultat : [ string ] }
SELECT CC2.Valeur.Adresse
FROM C IN LesClients, CC1 IN C.Coordonnées, CC2 IN C.Coordonnées
WHERE C.Nom = 'Alimentation du maquis'
AND CC1.Valeur.Adresse = 'Propriano'
AND CC2.Validite > CC1.Validite
AND NOT EXISTS CC3 IN C.Coordonnées :
        (CC3.Validité > CC1.Validité AND CC3.Validité < CC2.Validité)
```

A.6 Modélisation en TOOSQL

A.6.1 Schéma

TOOSQL est un langage de requêtes temporel défini dans le contexte d'un modèle à objets temporels et dont la syntaxe est proche de celle de SQL. Des concepts supportés par le modèle nous ne décrivons que ceux qui sont nécessaires à la compréhension des requêtes.

Le temps est associé aux attributs par l'intermédiaire de séquences temporelles (*time sequence, TS*). Un attribut historique est un attribut dont le type est une séquence temporelle paramétrée par le type de sa valeur structurelle : AH : TS (valeur : T).

Les séquences temporelles sont séquences de triplets < val, tv, tt > : l'attribut a pour valeur val pendant l'intervalle tv du monde réel et ceci est enregistré dans la base à l'instant tt. Les séquences temporelles sont des historiques bitemporels.

Le temps de transaction est fixé implicitement au moment où la requête est évaluée. Une clause particulière (la clause Rollback) permet de fixer explicitement cet instant.

Nous donnons ci-dessous le schéma de notre base de données exemple dont les historiques sont modélisés à l'aide de séquences temporelles.

Source : < Agrement: string, Debit: TS(real) >
{ $<a, Hb> \in Source \iff$ *la source a pour numéro d'agrément a et Hb est la séquence temporelle de l'historique de son débit* }

Bouteille : < Type: string, Volume: real, Production: TS(integer), Prix: TS(real) >
{ $< t, v, Ho, Hi > \in Bouteille \iff$ *la bouteille de type t a pour volume v, Ho est la séquence temporelle de l'historique de sa production et Hi est la séquence temporelle de l'historique de son prix* }

Camion : < Immat: string, DateAchat: date,
Entretien: TS(<Nature: string, Prix: real>),
Controle : TS(<Agrement: string, Remarques: string>) >
{ $< i, d, He, Hc > \in Camion \iff$ *le camion de numéro d'immatriculation i a été acheté à la date d et He est la séquence temporelle de l'historique de ses visites d'entretiens et Hc est la séquence temporelle de l'historique de ses contrôles techniques* }

Commande : < Numero: integer, Client: Client,
Lignes: [<TypeB: Bouteille, Quantite : integer>],
Camion : Camion >
{ $< n, c, l, o > \in Commande \iff$ *la commande de numéro n concerne le client c, est constitué des lignes de commandes l et sa livraison est effectuée par le camion o* }

Client : < Nom: string, Directeur: TS(string),
Coordonnées: TS(<Adresse: string, Téléphone: string>),
Commandes: TS(Commande) >
{ < *n, Hd, Ho, Hc* > ∈ *Client* ⟺ *pour le client de nom n, les séquences temporelles Hd, Ho, Hc correspondent aux historiques de ses directeurs, de ses coordonnées et de ses commandes* }

Dans ce modèle les noms de classe désignent aussi leur extension.

A.6.2 Requêtes

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

{ *type du résultat :* { < *string, date, [<string, real>]* > } }
SELECT C.Immat, C.DateAchat, C.Entretien
{ *l'expression C.Entretien construit la séquence des valeurs structurelles de l'attribut Entretien (une séquence temporelle) de l'objet C.* }
FROM C:Camion

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

{ *type du résultat :* { < *string, TS(<string, real>)* > } }
SELECT C.Immat, C.Entretien.History
{ *La méthode History appliquée à un attribut historique désigne sa séquence temporelle.* }
FROM C:Camion
WHERE C.Entretien.Nature ≠ 'vidange'

La condition de la clause WHERE exclut du résultat tous les éléments de la séquence temporelle qui ne correspondent pas à des vidanges.

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

{ *type du résultat :* { < *string, TS(prix)* > } }
SELECT B.Type, B.Prix.History
FROM B:Bouteille
TIME-SLICE [1/1/1990..1/12/1996]
{ *La clause TIME-SLICE restreint le résultat aux n-uplets valides pendant la période donnée* }
WHERE B.Prix.Duration(vt).Sum = Duration([1/1/1990..1/12/1996])
{ *La méthode Duration(vt) appliquée à un attribut historique construit les durées de chaque intervalle de la séquence temporelle associée. La méthode Sum effectue la somme de ces durées* }

Nous prenons ici pour hypothèse que la restriction temporelle introduite par la clause TIME-SLICE est effectuée avant l'évaluation de la condition de la clause WHERE.

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

```
{ type du résultat : { < string, TS(string) > } }
SELECT C.Nom, C.Directeur.History
FROM C:Client
WHEN C.Commandes.First ≤ '31/12/1994'
   { La clause WHEN introduit une sélection temporelle. }
   { La méthode First, appliquée à un attribut historique, donne la valeur temporelle la plus
   ancienne de la séquence temporelle correspondante }
TIME-SLICE (SELECT CC.Adresse.vt
            FROM CC:Client
            WHERE CC.Nom = C.Nom AND CC.Adresse = 'Paris')
```

La sous-requête construit l'ensemble des intervalles de temps pendant lesquels l'adresse du client est Paris. La clause TIME-SLICE permet de restreindre l'historique résultat à ces intervalles.

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

```
{ type du résultat : { instants } }
SELECT P.Production.vt
FROM P:Bouteille, Z:Bouteille
WHERE P.Type = 'PlusSoif' AND Z.Type = 'Zébu'
AND Z.Production.vt.Equals (P.Production.vt)
AND P.Production > Z.Production
UNION
SELECT P.Production.vt
FROM P:Bouteille
WHERE P.Type = 'PlusSoif'
WHEN P.Production.vt NOT IN (SELECT Z.Production.vt
                             FROM Z:Bouteille
                             WHERE Z.Type = 'Zébu')
```

Dans la deuxième sous-requête la clause WHEN permet de ne retenir que les instants où les bouteilles de type "PlusSoif" sont produites alors que celles de type "Zébu" ne le sont pas.

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

{ *type du résultat :* { < *string, TS(integer), TS(real)* > } }
SELECT B.Type, B.Production.History, B.Prix.History
FROM B:Bouteille
TIME-SLICE [1/1/997, 31/12/1997]

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

{ *type du résultat : integer* }
SELECT B.Production.Sum
FROM B:Bouteille
WHERE B.Type = 'PlusSoif'
AND B.Prix > 5

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

{ *type du résultat :* { < *TS(<string, real>)* > } }
SELECT C.Entretien.History
FROM C:Camion
WHERE C.Immat = '735 AOC 38'
WHEN C.Entretien.vt.Follows (SELECT CC.Entretien.vt.First
FROM CC:Camion
WHERE CC.Immat = C.Immat
AND CC.Entretien = 'vidange')
{ *La méthode Follows s'appliquent entre deux intervalles et déterminent si le premier est après le second.* }

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

```
{ type du résultat : { string } }
SELECT C.Adresse
FROM C:Client
WHERE C.Nom = 'Alimentation du maquis'
WHEN C.Adresse.vt.Follows (SELECT CC.Adresse.vt
                           FROM CC:Client
                           WHERE CC.Nom = C.Nom AND CC.Adresse = 'Propriano')
                           { Cette première condition permet de ne sélectionner que les
                           adresses après que le client ait quitté Propriano. }
AND NOT EXISTS (SELECT *
                FROM CCC:CLient
                WHERE CC.Nom = CCC.Nom
                WHEN CCC.Adresse.vt.Follows (C.Adresse.vt)
                AND CC.Adresse.vt.Follows (CCC.Adresse.vt))
                { Cette seconde condition permet de ne retenir que les adresses
                correspondant à l'adresse où à déménager le client, juste après avoir quitté
                Propriano. }
```

A.7 Modélisation en Tempos

A.7.1 Schéma

Les classes sont obtenues directement issues de la modélisation OMT.
TEMPOS propose différents statuts d'évolution pour les attributs (Cf. section 3.1) :

- Constant (T) : la valeur de l'attributΓde type TΓn'évolue pas.
- Fugitif (T) : pour l'attribut de type TΓseule la valeur de la dernière mise à jour est stockée.
- Historique (u,T) : l'attribut est de type TΓses valeurs sont historisées et les estampilles temporelles sont de granularité u.

La définition des classes TEMPOS est la suivante :

```
class Source inherit Object
public type tuple (
    Agrement : Constant (string),
    Debit : Historique (heure, real))
end ;

class Bouteille inherit Object
public type tuple (
    Type : Constant (string),
    Volume : Fugitif (real),
    Production : Historique (jour, integer),
    Prix : Historique (jour, real))
end ;

class Camion inherit Object
public type tuple (
    Immat : Constant (string),
    DateAchat : Fugitif (Instant (jour)),
    Entretien : Historique (jour,
            tuple(Nature: string, Prix: real)),
    Controle : Historique (mois,
                    tuple(Agrement: string,
                        Remarques: string)))
end ;
```

```
class Commande inherit Object
public type tuple (
    Numero : Constant (integer),
    Client : Fugitif (Client),
    Détail : Fugitif (list (tuple (
                        TypeB: Bouteille,
                    Quantite: integer)))
    Livraison : Fugitif (Camion))
end ;

class Client inherit Object
public type tuple (
    Nom : Constant (string),
    Directeur : Historique (Jour, string),
    Coordonnées : Historique (Jour,
     tuple(adresse: string, telephone: string)),
    Commandes : Historique (jour, Commande))
end ;

name LaSource : Source ;
name LesBouteilles : set (Bouteille) ;
name LesCamions : set (Camion) ;
name LesClients : set (Client) ;
name LesCommandes : set (Commande) ;
```

A.7.2 Requêtes

Les opérations de consultation de TEMPOS sont définies dans la section 3.2 et les fonctions de manipulation de séquences le sont dans l'annexe C.

R.1 *Pour chaque camion, donner son numéro, sa date d'achat et l'ensemble des visites d'entretiens qu'il a subi (nature et coût).*

{ *type du résultat :* {<*numero : string, dateach : instant, ent :* { *tuple(string, real)* } >} }
SELECT TUPLE (numero : c.Immat, dateach : c.DateAchat, ent : DomS(c.Entretien))
FROM c IN LesCamions

R.2 *Pour chaque camion, donner son numéro et l'historique de ses visites d'entretien (nature et coût) lorsqu'il ne s'agissait pas de vidanges.*

{ *type du résultat :* {<*immat : string, ent : Chronique (mois, tuple(string, real))*>} }
SELECT TUPLE (immat : c.Immat,
ent : c.Entretien $\lceil_{si}$ λv • v.Nature $\neq$ 'Vidange'
FROM c IN LesCamions
WHERE (c.Entretien $\lceil_{si}$ λv • v.Nature $\neq$ 'Vidange') $\neq \emptyset$

R.3 *Pour chaque type de bouteille en vente pendant tout l'intervalle [1/1/1990..1/12/1996], donner son nom et l'historique de ses prix, restreint à cet intervalle.*

{ *type du résultat :* {<*type : string, prix : Chronique (jour, real)*>} }
SELECT TUPLE (type : b.Type,
prix : b.Prix $\lceil_{en}$ [@'1-1-1990'..@'1-12-1996'])
FROM b IN LesBouteilles
WHERE (b.Prix $\lceil_{en}$ [@'1-1-1990'..@'1-12-1996']) $\neq \emptyset$

R.4 *Pour chaque client (connu depuis 1994), donner son nom et l'historique de ses directeurs lorsque ce client se trouvait situé à Paris.*

{ *type du résultat :* {<*nom : string, dir : Chronique (jour, string)*>} }
SELECT TUPLE (nom : c.Nom,
dir : c.Directeur $\lceil_{en}$ LesInstants (c.Coordonnees, λa • a.adresse = 'Paris'))
FROM c IN LesClients
WHERE DomT (c.Commandes)$^-$ $\leq$ (@'31-12-1994')
{ *L'historique résultat est vide pour les clients n'ayant pas été à Paris.* }

R.5 *Quand les bouteilles de type "PlusSoif" ont-elles été produites en plus grande quantité que les bouteilles de type "Zébu" ?*

{ *type du résultat : {Instant (jour)}* }
SELECT LesInstants ((p.Production $*_{\cap}$z.Production), λ<pp, pz> • pp > pz)
FROM p IN LesBouteilles, z in LesBouteilles
WHERE p.Type = 'PlusSoif' and z.Type = 'Zébu'

R.6 *Pour chaque type de bouteille vendu et produit pendant 1997, donner son nom et l'historique, restreint à cette période, de sa production et de son prix.*

{ *type du résultat : {<type : string, prixprod : Chronique (jour, <integer, real>)>}* }
SELECT TUPLE (type : b.Type,
prixprod : ((b.Production $\lceil_{si}$ Est_non_Nil) $*_{\cap}$(b.Prix $\lceil_{si}$ Est_non_Nil))
$\lceil_{en}$ [@'1/1/1997', @'31/12/1997'])
FROM b IN Les Bouteilles

R.7 *Combien a-t-on produit de bouteilles de type "PlusSoif" lorsqu'elles coûtaient plus de 5 francs ?*

{ *type du résultat : integer* }
SELECT Agrégation (I_Chr (b.Production $\lceil_{en}$ LesInstants (b.Prix, λx • x > 5)),
0, λacc, i • acc + VS (i))
FROM b IN LesBouteilles
WHERE b.Type = 'PlusSoif'

R.8 *Quel est l'historique des entretiens faits sur le camion immatriculé "735 AOC 38" depuis sa première vidange ?*

{ *type du résultat : X_chronique (jour, tuple(string, real))* }
SELECT LaFin (c.Entretien, λx • VS (x).Nature = 'Vidange')
FROM c in LesCamions
WHERE c.Immat = '735 AOC 38'

R.9 *A quelle(s) adresse(s) a déménagé le client "Alimentation du maquis" juste après qu'il ait quitté Propriano ?*

{ *type du résultat : [string]* }
SELECT Application (
Sélection (LesCouples (X_Hist (c.Coordonnees)),
λ<i1, i2> • VS(i1).Adresse = 'Propriano' $\wedge$ VS(i2).Adresse $\neq$ 'Propriano'),
λ<i1, i2> • VS (i2.Adresse)
FROM c IN LesClients
WHERE c.Nom = 'Alimentation du maquis'

Annexe B

Application : séries chronologiques

L'application du modèle TEMPOS aux séries chronologiques a été fait en collaboration avec Marie-Claude Quidoz dans le cadre de son mémoire d'ingénieur CNAM [Qui97].

B.1 Position du problème

B.1.1 Séries chronologiques

Une série chronologique est une suite de valeurs dans le temps. Les séries chronologiques sont utilisées en économie pour représenter des statistiques sur des entités évoluant dans le temps, comme par exemple, la population, un parc de voiture, la consommation de pétrole, etc. La figure B.1 est un exemple de série sur le parc automobile en Turquie.

Une série chronologique est caractérisée par :

- Un identifiant,
- Un ensemble de caractéristiques générales, comme la source de la série, sa date de saisie, l'unité des valeurs, son degré de fiabilité, etc.,
- Un calendrier définissant le domaine temporel de la série,
- Une liste de valeurs, éventuellement annotées.

La liste des valeurs correspond à l'historique des valeurs du domaine de la série. Certaines valeurs peuvent ne pas être diffusées (pour raison de confidentialité par exemple). Dans ce cas, la valeur est notée `na` (non accessible).

Identifiant	car.tur.u.irf.1977
Date élaboration	15/03/1996
Source	irf
Date parution	1977
Sujet	Parc automobile
Zone géographique	Turquie
Unité	Véhicules
Observation	
Périodicité	Annuelle
Date de début	1972

Données

Date	Valeur	Annotation
1972	185 266	
1973	234 577	estimation
1974	303 845	
1975	419922	
1976	512 380	

FIG. B.1 – *Exemple de série chronologique*

B.1.2 Différents types de séries

Il existe trois catégories de séries chronologiques :

- Les *séries brutes* sont diffusées par les producteurs de données et sont intégrées telles quelles dans le système de gestion des séries (que ce soit un SGBD ou un logiciel spécifique).

- Les *séries unifiées* correspondent à une première étape de modification des séries mais les données ne sont pas interprétées. Il s'agit plutôt de modification de forme comme le changement de périodicité (cumul des mois pour passer en années) d'unité (passer de miles^2 en km^2) voire la fusion de deux séries de domaines temporels différents (par exemple passer d'une série de 1970 à 1980 et d'une série de 1981 à 1990 à une seule série de 1970 à 1990). Dans tous les cas les opérations réalisées dans ce cadre correspondent à des opérations bien précises.

- Les *séries dérivées* sont construites à partir de séries unifiées ou d'autres séries dérivées à l'aide d'équations mathématiques. La validation des modèles utilisés est faite par un expert.

B.1.3 Modélisation OMT des séries

La figure B.2 montre pour la structuration des séries et les liens qui existent entre les différentes séries.

Quelle que soit le type d'une série, celle-ci est composée d'un descriptif (caractérisant la série) et de données (représentant l'historique des valeurs). La structure des descriptifs et des données est décrit dans la figure B.3

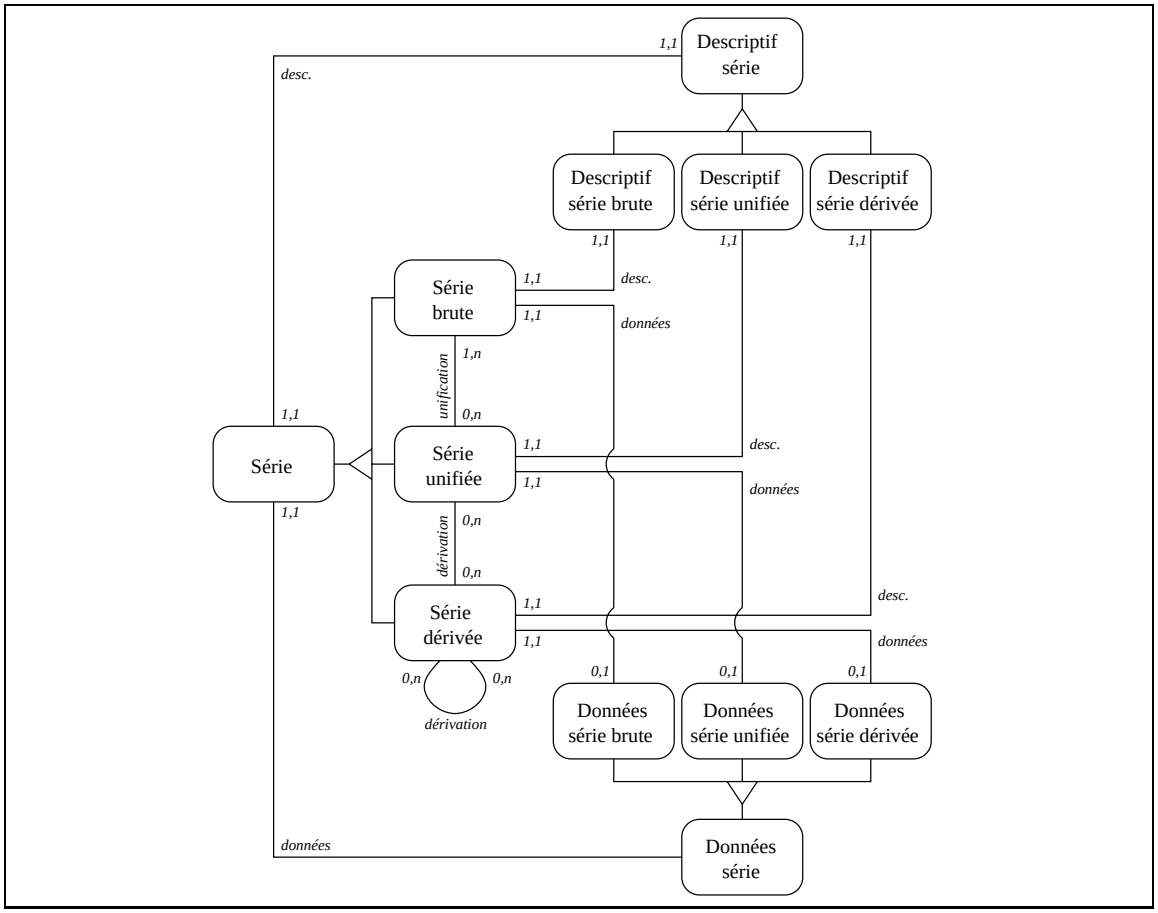
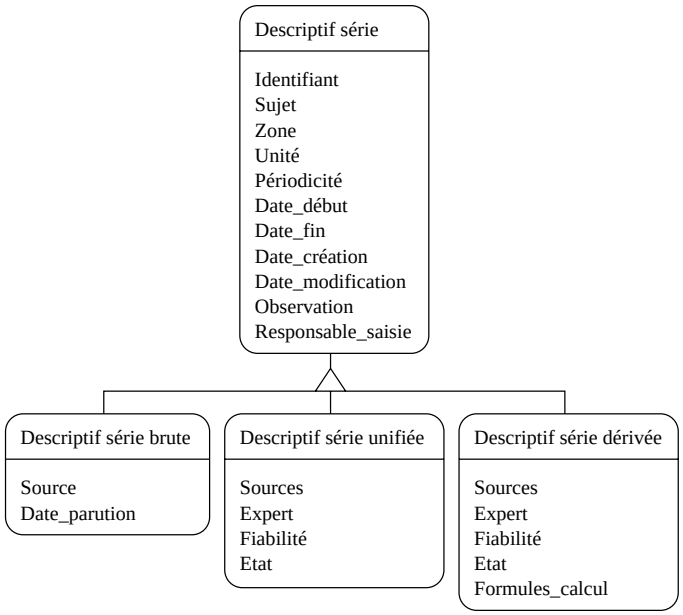
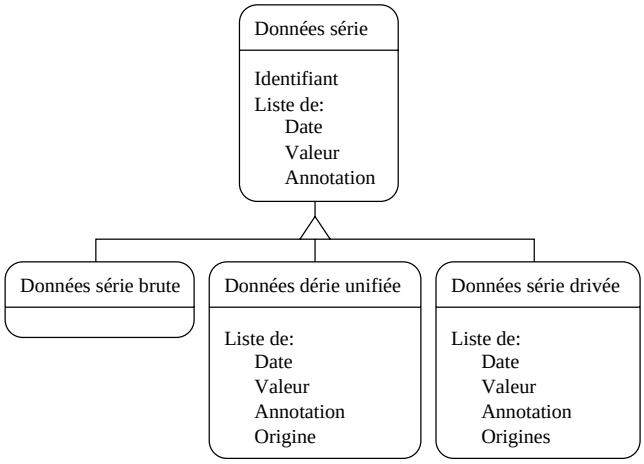


FIG. B.2 – *Modélisation OMT des séries*



(a) Structure des descriptifs



(b) Structure des données

FIG. B.3 – *Structure des descriptifs et des données de séries*

B.2 Séries chronologiques et Tempos

B.2.1 Représentation des séries dans Tempos

Les valeurs saisies d'une série brute forment une chronique. Les valeurs inconnues (notées na) sont explicitement saisies par l'utilisateur. Il s'agit de valeurs effectives : l'utilisateur sait que cette valeur est inaccessible.

La présentation habituelle des séries se fait à domaine temporel convexe, aussi, pour masquer les discontinuités éventuelles de la saisie, il est possible de modéliser une série brute à l'aide d'un historique discret, ayant un domaine temporel allant de la première à la dernière valeur effective. De cette manière, les valeurs non saisies seront interprétées comme des valeurs absentes (Cf. figure B.4).

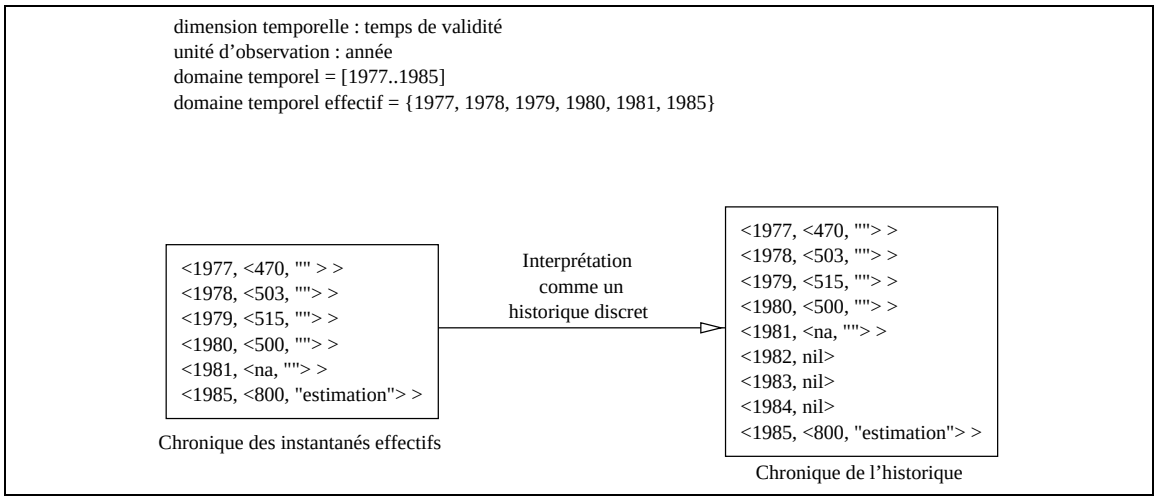


FIG. B.4 – *Représentation d'une série brute*

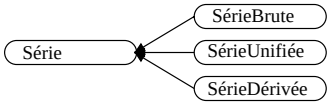
B.2.2 Extension de Tempos

Afin de modéliser les séries chronologiques dans TEMPOS, nous ajoutons de nouveaux types, définis dans la figure B.5.

Les types Donnée, DonnéeBrute, DonnéeUnifiée, DonnéeDérivée permettent de décrire les données des diverses séries.

De même Descriptif (u), DescriptifBrut (u), DescriptifUnifié (u) et DescriptifDérivé (u) correspondent aux caractéristiques générales des séries. Ces types sont paramétrés par une unité d'observation du temps qui correspond habituellement à la périodicité des séries. Les attributs Instant(Jour) correspondent aux dates de création et de modification qui sont observées à la granularité du jour.

Les informations temporelles habituellement enregistrées dans les séries (dates de début et de fin des valeurs dans le descriptif, date de chaque valeur dans les données) ne



(a) Hiérarchie des types

TYPE DescriptifBrut(u) : < Sujet, Zone, Unité, u, Instant(Jour), Instant(Jour), texte, Personne, Source, Instant(jour) >
{ *Descriptif caractérisant une série brute* }

TYPE DescriptifUnifié(u) : < Sujet, Zone, Unité, u, Instant(Jour), Instant(Jour), texte, Personne, [Source], Personne, texte, texte >
{ *Descriptif caractérisant une série unifiée* }

TYPE DescriptifDérivé(u) : < Sujet, Zone, Unité, u, Instant(Jour), Instant(Jour), texte, Personne, [Source], Personne, texte, texte, [< Intervalle(u), texte >] >
{ *Descriptif caractérisant une série dérivée* }

TYPE Descriptif(u) : DescriptifBrut(u) ∪ DescriptifUnifié(u) ∪ DescriptifDérivé(u)

TYPE DonnéeBrute : < réel, texte >
{ *Les données des séries brutes sont des couples <valeur, annotation>* }

TYPE DonnéeUnifiée : < réel, texte, texte >
{ *Les données des séries brutes sont des triplets <valeur, annotation, origine>* }

TYPE DonnéeDérivée : < réel, texte, [texte] >
{ *Les données des séries brutes sont des triplets <valeur, annotation, liste d'origines>* }

TYPE Donnée : DonnéeBrute ∪ DonnéeUnifiée ∪ DonnéeDérivée

TYPE Série(u) : < Descriptif(u), HistDiscret(u, Donnée) >
{ *Une série chronologique à l'unité u est constituée d'un descriptif et d'un historique discret de données observé à l'unité u* }

TYPE SérieBrute(u) : < DescriptifBrut(u), HistDiscret(u, DonnéeDrute) >
{ *Spécialisation pour les séries brutes* }

TYPE SérieUnifiée(u) : < DescriptifUnifié(u), HistDiscret(u, DonnéeUnifiée) >
{ *Spécialisation pour les séries unifiées* }

TYPE SérieDérivée(u) : < DescriptifDérivé(u), HistDiscret(u, DonnéeDérivée) >
{ *Spécialisation pour les séries dérivées* }

(b) Description des types relatifs aux séries

FIG. B.5 – *Types de* TEMPOS *pour les séries*

sont plus représentées explicitement dans les types de TEMPOS elles sont déduites de l'historique.

Enfin les types Série Série Brute Série Unifiée et Série Dérivée représentent les différents types de séries chronologiques. Elles sont constituées d'un descriptif et d'un historique discret de données observé dans l'unité correspondant à leur périodicité.

B.2.3 Exemples de requêtes sur les séries

Nous supposons ici disposer d'une base où se trouvent des données relatives à un grand nombre de séries. L'accès à ces données se fait par la racine de persistance LesSeries.

1. Valeur des séries relatives au prix du pétrole entre 1980 et 1990 :

```
SELECT TUPLE (serie: S.Identifiant,
              val8090: S.Données ⌈en [@'1980'..@'1990'])
FROM S IN LesSeries
WHERE S.Descriptif.Sujet = 'prix du pétrole'
```

2. Valeur des séries relatives au prix du pétrole depuis 1985 :

```
SELECT TUPLE (serie: S.Identifiant,
              val85: LaFin(S.Données, λvt • vt = @'1985')
FROM S IN LesSeries
WHERE S.Descriptif.Sujet = 'prix du pétrole'
```

3. Date de la première valeur de la série identifiée par pétrole.fr :

```
SELECT DomT(S.Données)⁻
FROM S IN LesSeries
WHERE S.Descriptif.Identifiant = 'pétrole.fr'
```

4. Années pour lesquelles les valeurs de la série identifiée par pétrole.fr sont supérieure à 100 francs/baril :

```
SELECT LesInstants (S.Données, λvs • vs.Valeur > 100)
FROM S in LesSeries
WHERE S.Descriptif.Identifiant = 'pétrole.fr'
```

5. Années pour lesquelles les valeurs de la série identifiée par pétrole.fr sont supérieures à celles de la série identifiée par gaz.fr :

{ *Nous ne considérons ici que les années où les deux valeurs sont connues et accessibles (différentes de* na*). Pour cela, nous éliminons d'abord les valeurs* na *et ensuite nous faisons un produit interne entre les deux historiques* }

```
SELECT LesInstants ((S1.Données ⌈si λvs • vs.Valeur ≠ na
                    *∩S2.Données ⌈si λvs • vs.Valeur ≠ na),
                    λ<vs1, vs2> • vs1.Valeur > vs2.Valeur)
FROM S1 IN LesSeries, S2 IN LesSeries
WHERE S1.Descriptif.Identifiant = 'pétrole.fr'
AND S2.Descriptif.Identifiant = 'gaz.fr'
```

B.2.4 Bilan

Les séries chronologiques sont un cas bien particulier d'historiques. Les données sont périodiques (presque toujours à la même unité)Γle domaine temporel est un intervalle.

Nous n'avons pas mis en oeuvre dans cette application toute les fonctionnalités relatives au modèle du temps (manipulation de différentes valeurs temporelleΓmultigranularitéΓetc.).

Par contreΓnous avons utilisé le modèle d'historiques pour représenter les différents types de séries chronologiques. TEMPOS est un modèle très généralΓson utilisation pour les séries n'a donc pas posé de difficulté. L'expression de requêtes significatives dans le domaine se fait sans problème avec les opérateurs fournis.

L'utilisation des valeurs non accessibles est un besoin spécifique aux séries chronologiques. En tant que telΓil est pris en charge par l'application. Il s'agit du problème général lié aux interprétations multiples des valeurs absentes. Ces interprétations dépendent fortement du domaine concerné et ne peuvent être intégrées dans le modèle.

L'application de TEMPOS aux séries chronologiques permet de valider le modèle sur un cas concret. Il serait intéressant de prolonger cette collaboration avec Marie-Claude Quidoz et l'IEPE pour approfondir cette expérimentation et la mener en vraie grandeur.

Annexe C

Manipulation de séquences

Nous décrivons ici les notations concernant les séquences et nous définissons un ensemble de fonctions permettant de les manipuler [SFLM93].

C.1 Constructeurs et sélecteurs de base

T étant un typeΓ[T] dénote le type séquence de T. La séquence vide est notée []. Elle est caractérisée par le prédicat EstVide?. L'opérateur de concaténation de deux séquences est noté &. De plusΓune séquence peut être construite par ajout d'un élément à droite ou à gauche d'une séquence :

- Au constructeur d'ajout à gaucheΓnoté $_\circ$Γsont associés les sélecteurs nommés premier et fin (tout sauf le premier) :

  premier($e_\circ S$) = e
  fin($e_\circ S$) = S
  ¬EstVide?(S) ⇒ premier(S)$_\circ$fin(S) = S

- Au constructeur d'ajout à droiteΓnoté •Γsont associés les sélecteurs nommés dernier et début (tout sauf le dernier) :

  début(S•e) = S
  dernier(S•e) = e
  ¬EstVide?(S) ⇒ début(S)•dernier(S) = S

C.2 Fonctions sur les séquences

Nous définissons ici quelques itérateurs sous forme fonctionnelle par récurrence (T et T' dénotent des types).

– Application d'une fonction f aux éléments d'une séquence.

Application : [T], (T $\longrightarrow$ T') $\longrightarrow$ [T']
{ *Application ([e_1, e_2, .., e_n], f) = [f (e_1), f (e_2), .., f (e_n)]; Application ([ ], f) = [ ]* }
(1) Application([], f) = []
(2) Application($e_\circ$S, f) = f(e)$_\circ$Application (S, f)

– Sélection : sélection des éléments d'une séquence vérifiant un prédicat P.

Sélection : [T], (T $\longrightarrow$ booléen) $\longrightarrow$ [T]
{ *Sélection (S, P) = [e | e $\in$ S $\wedge$ P (e)]* }
(1) Sélection([], P) = []
(2) Sélection($e_\circ$S, P) = (si P(e) alors [e] sinon []) & Sélection(S, P)

– Sélection et Application : application d'une fonction f aux éléments vérifiant une propriété P.

Appli_Sel : [T], (T $\longrightarrow$ booléen), (T $\longrightarrow$ T') $\longrightarrow$ [T']
{ *Appli_Sel(S, P, f) = Application(Sélection(S, P), f)* }
(1) Appli_Sel([], P, f) = []
(2) Appli_Sel($e_\circ$S, P, f) = (si P(e) alors [f(e)] sinon []) & Appli_Sel(S, P)

– Agrégation : une fonction f est appliquée de manière cumulative aux éléments d'une séquenceΓde gauche à droite.

Agrégation : [T], T', (T',T $\longrightarrow$ T') $\longrightarrow$ T'
{ *Agrégation ([e_1, e_2, .., e_n], B, Op) = Op (Op (..Op (Op (B, e_1), e_2), ...), e_n); Agrégation ([ ], B, OP) = B.* }
(1) Agrégation ([ ], B, f) = B
(2) Agrégation (S•e, B, f) = f(Agrégation(S, B, f), e)

– Les couples : liste des couples successifs de la séquence.

LesCouples : [T] $\longrightarrow$ [<T, T>]
{ *LesCouples ([e_1, e_2, .., e_n]) = [<e_1, e_2>, <e_2, e_3>, .., <e_{n-1}, e_n>]*
LesCouples ([e_1]) = [ ]; LesCouples ([ ]) = [ ] }
(1) LesCouples ([ ]] = [ ]
(2) LesCouples ([e]) = [ ]
(3) LesCouples ([e_1, e_2]&S) = <e_1, e_2>$_\circ$LesCouples ($e_{1\circ}$S)

– Découpage d'une séquence selon une propriété : une séquence est découpée en deux parties selon une propriété P. Le point de séparation est le premier élément vérifiant P (de gauche à droite).

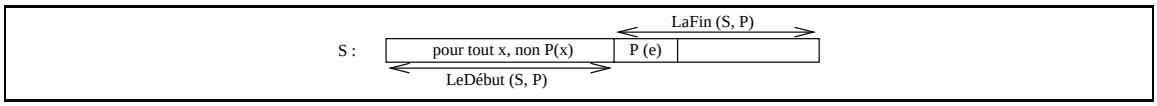


FIG. C.1 – *Découpage d'une séquence selon une propriété*

LeDébut : [T], (T ⟶ booléen) ⟶ [T]

{ *LeDébut (S, P) est la séquence de début de S comprise entre le premier élément de S (inclus) et le premier élément de S vérifiant P (exclu). LeDébut (S, P) est S si S ne comporte aucun élément vérifiant P.* }

(1) LeDébut([], P) = []

(2) LeDébut($e_\circ$S, P) = si P(e) alors [] sinon $e_\circ$LeDébut(S, P)

LaFin : [T], (T ⟶ booléen) ⟶ [T]

{ *LaFin (S, P) est la séquence de fin de S comprise entre le premier élément de S (inclus) vérifiant P et le dernier élément de S. LaFin (S, P) est vide si S ne comporte aucun élément vérifiant P.* }

(1) (1)LaFin([], P) = []

(2) LaFin($e_\circ$S, P) = si P(e) alors $e_\circ$S sinon LaFin(S, P)

DebEtFin : [T], (T ⟶ booléen) ⟶ <[T], [T]>

{ *DebEtFin(S, P) = <LeDébut(S,P), LaFin(S, P)>* }

Bibliographie

[AC93] M. Adiba et C. Collet. *Objets et bases de données : le SGBD O2.* HermèsΓ1993.

[ADF+94] T. AtwoodΓJ. DuhlΓG. FerranΓM. Loomis et D. Wade. *The Object Database Standard : ODMG-93, R.G.G. Cattell Ed.* Morgan KaufmannΓ1994.

[Adi96] M. Adiba. STORM an object-orientedΓmultimedia DBMS. Dans K.NowsuΓ éditeurΓ*Multimedia Database Management Systems*Γchapitre 3Γpages 47–88. Kluwer Academic PublishersΓ1996.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*Γ26(11)ΓNovember 1983.

[ANC87] M. AdibaΓB. Q. Ngoc et C. Collet. Aspects temporelsΓhistoriques et dynamiques dans les bases de données. *T.S.I., Numéro spécial Bases de données*Γ6(5):457–478Γ1987.

[Ari87] G. Ariav. Design requirement for temporally oriented information systems. In TAI87 [TAI87].

[ASU86] A. AhoΓR. Sethi et J. Ullman. *Compilers : principles, techniques and tools.* Addison-WesleyΓ1986.

[BBJS97] J. BairΓM. BöhlenΓC.S. Jensen et R.T. Snodgrass. Notions of upward compatibility of temporal query languages. Rapport de recherche TR-7ΓTime CenterΓ 1997.

[BCTP95] V. BrusoniΓL. ConsoleΓP. Terenziani et B. Pernici. Extending temporal relational databases to deal with imprecise and qualitative temporal information. Dans J. Clifford et A. TuzhilinΓéditeursΓ*proc. of the workshop on Recent Advances in Temporal Databases*ΓWorkshops in ComputingΓZurichΓSwitzerlandΓ September 1995. Springer Verlag.

[BDA96] *actes des 12e Journées Bases de Donnéees Avancées*ΓCassis (France)Γaoût 1996.

[Ben94] B. Benatallah. Evolution de schéma et systèmes à objets : synthèse. Dans *actes du congrès INFORSID*ΓAix-en-ProvenceΓFranceΓMai 1994.

[BF97] B. Benatallah et M.-C. Fauvet. Le point sur l'évolution du schéma d'une base d'objets. *L'OBJET: logiciels, bases de données, réseaux*Γ1997. 10 pagesΓà paraître.

[Böh94] M. Böhlen. *Managing temporal knowledge in deductive databases.* PhD thesisΓETH-ZurichΓ1994.

[Böh95] M.H. Böhlen. Temporal database system implementations. *SIGMOD record*Γ24(4):53–60ΓDecember 1995.

[BL89] H. Bestougeff et G. Ligozat. *Outils logiques pour le traitement du temps, de la linguistique à l'intelligence artificielle.* MassonΓ1989.

[BSS96] M. BöhlenΓR. Snodgrass et M. Soo. Coalescing in Temporal Databases. Dans *proc. of the 22nd VLDB Conference*ΓMumbai (Bombay)ΓIndiaΓ1996.

[BWJ96] C. BettiniΓX. Sean Wang et S. Jajodia. A general framework and reasoning models for time granularity. Dans *proc. of the 3rd Conference on Temporal Representation and Reasoning*ΓKey WestΓFLΓ1996.

[Can96a] J.-F. Canavaggio. Types temporels pour un SGBD à objets. Dans *actes de Journées Jeunes Chercheurs MATIS*ΓArchampsΓFranceΓdécembre 1996.

[Can96b] J.-F. Canavaggio. Une bibliothèque temporelle. Dans *Journée O_2 dans le cadre du Congrès INFORSID'96*ΓBordeauxΓjuin 1996.

[Can96c] J.-F. Canavaggio. Une bibliothèque temporelle pour un SGBD. Dans *actes des deuxièmes journées des jeunes chercheurs, GDR - PRC Bases de données - Pôle modélisation et environnements de développement*Γpages 117–130ΓParisΓjanvier 1996.

[Cas93] X. Castellani. *MCO, Méthodologie générale d'analyse et de conception des systèmes d'objets.* Masson (Paris)Γ1993.

[CC93] J. Clifford et A. Croker. The Historical Relational Data Model (HRDM) revisited. In Tansel et al. [TCG+93].

[CC96] C. Collet et T. Coupaye. Primitive and Composite Events in NAOS. In *actes des 12e Journées Bases de Donnéees Avancées* [BDA96]. pages 331–349.

[CCMP93] E. CiapessoniΓE. CorsettiΓA. Montanari et P. San Pietro. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of computer programming*Γ20:141–171Γ1993.

[CD95] D. Cukierman et J. Delgrande. A language to express time intervals and repetition. Dans *proc. of the TIME-95 International Workshop on Temporal Representation and Reasoning*ΓMelbourneΓAustraliaΓ1995.

[CD97] J.-F. Canavaggio et M. Dumas. Manipulation de valeurs temporelles dans un SGBD à objets. Dans *actes du 15e congrès INFORSID*ΓToulouseΓjuin 1997.

[CLR89] C. CauvetΓJ.Y. Lingat et C. Rolland. Information system engineering : the Rubis system. Dans *proc. of the International Conference on CASE tools (CAISE)*Γ StockholmΓSwedenΓ1989.

[CO93] S. J. Cannan et G. A. M. OttenΓéditeurs. *SQL - The standard handbook.* McGraw-Hill Book CompagnyΓ1993.

[Cou86] P. Couderc. *Le Calendrier.* Que Sais-Je. Presses Universitaires de FranceΓ1986.

[CR87] J. Clifford et A. Rao. A simple general structure for temporal domains. In TAI87 [TAI87].

[CSS94] R. ChandraΓA. Segev et M. Stonebraker. Implementing calendars and temporal rules in the next generation databases. Dans *proc. of the International Conference on Data Engineering, IEEE*Γ1994.

[DBS96] D. DeyΓT. M. Barron et V. C. Storey. A complete temporal relational algebra. *The VLDB Journal*Γ5(3)ΓAugust 1996.

[Deu90] O. Deux. The story of O_2. *IEEE Transaction on Knowledge and Data Engineering*Γ2(1)ΓMarch 1990.

[DFG+96] A. DoucetΓM.-C. FauvetΓS. GançarskiΓG. Jomier et S. Monties. Using database versions to implement temporal integrity constraints. Dans *International Workshop on constraints in databases, LNCS No1191*ΓBostonΓ USAΓAugust 1996. 10 pages.

[DLW84] P. DadamΓV. Lum et H. D. Werner. Integration of time versions into a relational databases system. Dans *proc. of 10th VLDB International Conference*Γ SingapourΓAug 1984.

[Dum96] M. Dumas. Vers un langage d'expressions temporelles. rapport de stage de Magistère de Mathématiques Fondamentales et Appliquées et d'InformatiqueΓ ENS de ParisΓseptembre 1996.

[Dum97] M. Dumas. Expression de contraintes d'intégrité dans un SGBD Temporel à objets. Rapport de rechercheΓDEA Systèmes et CommunicationsΓUniversité

Joseph Fourier – Institut National Polytechnique de GrenobleΓjuin 1997. 100 pages.

[EPP93] N. EdelweissΓJ. Palazzo et B. Pernici. An object-oriented temporal model. Dans *proc. of the CAISE'93 Conference*ΓParis (France)Γ1993. Springer Verlag. LNCS 685.

[EPP94] N. EdelweissΓJ. Palazzo et B. Pernici. An object-oriented approach to a temporal query language. Dans *proc. of the 5th International Conference on Database and Expert Systems and Applications (DEXA), LNCS 856*ΓAthensΓGreeceΓ 1994. Springer Verlag.

[Euz94] J. Euzenat. Granularité dans les représentations spatio-temporelles. Rapport de recherche N2242ΓINRIAΓGrenobleΓavril 1994.

[Fau89] M. C. Fauvet. Définition et réalisation d'un système de gestion de versions d'objets. Dans *5èmes Journées de Bases de Données Avancées*ΓGenèveΓSuisseΓ Septembre 1989.

[Fau92] M. C. Fauvet. Versions and histories in object-oriented applications. Dans *7SBBD, the 7th Bresilian Conference on Databases*ΓPorto-AlegreΓBrazilΓMay 13-15 1992.

[Fay97] L. Fayolle. Définition et détection d'événements dans NAOS. Mémoire CNAM en InformatiqueΓjuin 1997.

[FCS96] M.-C. FauvetΓJ.-F. Canavaggio et P.-C. Scholl. Expressions de requêtes temporelles dans un SGBD à objets. In *actes des 12e Journées Bases de Donnéees Avancées* [BDA96]Γpages 225–250.

[FCS97a] M.-C. FauvetΓJ.-F. Canavaggio et P.-C. Scholl. Modelling histories in object DBMS. Dans *proc. of the 8th International Conference on Database and Expert Systems Applications (DEXA)*Γ Toulouse (France)Γ September 1997. Springer Verlag. LNCS 1308.

[FCS97b] M.-C. FauvetΓJ.-F. Canavaggio et P.-C. Scholl. TEMPOS : un modèle d'historiques pour un *SGBD* temporel à objets. Dans *actes des 13e journées de Bases de Données Avancées*ΓGrenoble (France)Γseptembre 1997.

[FS95] M.-C. Fauvet et P.-C. Scholl. Temps et bases de données : concepts temporels pour la gestion de l'évolution des données. Rapport de recherche RR 945 IΓ Laboratoire de Génie InformatiqueΓIMAGΓmars 1995.

[Gad93] S. Gadia. Ben-zvi's pioneering work in relational temporal databases. In Tansel et al. [TCG+93].

[Gir95] J.-P. Giraudin. Evolution de la modélisation des systèmes d'information. Dans *proc. of the 8th International Conference on Software Engineering and its Applications*ΓParisΓnovember 1995.

[GLOS97] I. A. GoralwallaΓY. LeontievΓT. Ozsu et D. Szafron. Modeling temporal primitives ; back to basics. Dans *proc. of CIKM Conference*ΓLas VegasΓNovember 1997.

[GN93] S. K. Gadia et S. S. Nair. Temporal databases: a prelude to parametric data. In Tansel et al. [TCG+93].

[GO93] I. A. Goralwalla et M. T. Ozsu. Temporal extensions to a uniform behavioral object model. Dans *proc. of the 12th International Conference on the Entity-Relationship Approach - ER'93, LNCS 823*. Springer VerlagΓ1993.

[GV85] S. K. Gadia et J. H. Vaishnav. A query language for a homogeneous temporal database. Dans *proc. of Conf on Principles of Database Systems*ΓApril 1985.

[Ha91] J.-P. Haton et al. *Le raisonnement en Intelligence Artificielle*. InterEditionsΓ 1991.

[Hob85] J. R. Hobbs. Granularity. Dans *proc. of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*Γ1985.

[Ins92] American National Standards InstituteΓéditeur. *American National Standard for Information Systems - Database Language - SQL. ANSI X3, 135-1992. Revision and consolidation of ANSI X3, 135-1989 and ANSI X3, 168-1989*. 1992.

[ISO88] International Standards OrganizationΓéditeur. *ISO 8601:1988 : Data elements and interchange formats – Information interchange – Representation of dates and times*. 1988. 14 pages.

[JCE+94] C.-S. JensenΓJ. CliffordΓR. ElmasriΓS. GadiaΓP. Hayes et S. Jajodia. A consensus glossary of temporal database concepts. *ACM SIGMOD Record*Γ23(1)Γ March 1994.

[JSS93] C. JensenΓM. Soo et R. Snodgrass. Unification of temporal data models. *Data Engineering Conference, IEEE*Γ1993.

[Kaf93] W. Kafer. Temporal selectionΓtemporal projection and temporal join revisited. In Snodgrass [Sno93].

[KJ93] B. Knight et M. Jixin. An extended temporal system on points and intervals. *Information Systems*Γ18(2):111–120Γ1993.

[Kou94] M. Koubarakis. Database models for infinite and indefinite temporal information. *Information Systems*Γ19(2):141–173Γ1994.

[KS92] W. Kafer et H. Schoning. Realizing a temporal complex-object data model. Dans *proc. of the ACM SIGMOD Conference*ΓCaliforniaΓJune 1992.

[Lad86] P. Ladkin. Time representation: A taxonomy of interval relations. Dans *proc. of the 5th National Conference on Artificial Intelligence (AAAI'86)*ΓPhiladelphiaΓ 1986.

[LBG93] T. LawsonΓC. Balthazaar et A. Gray. An object-oriented approach to temporal modelling. In Snodgrass [Sno93].

[LEW96] J. Y. LeeΓR. Elmasri et J. Won. Specification of calendars and time series for temporal databases. Dans *proc. of International Conference on Conceptual Modeling - ER'96*ΓCottbusΓGermanyΓOctober 1996. Springer VerlagΓLNCS No 1157.

[LJ88] N. Lorentzos et R. Johnson. Extending relational algebra to manipulate temporal data. *Information Systems*Γ13(3)Γ1988.

[LMF86] B. LebanΓD. McDonald et D. Forster. A representation for collection of temporal intervals. Dans *proc. of the 5th National Conference on Artificial Intelligence (AAAI'86)*ΓPhiladelphiaΓ1986.

[Lor93] N. A. Lorentzos. The Interval-extended Relational Model and its application to valid-time databases. In Tansel et al. [TCG+93].

[Mac86] E. MacKenzie. Bibliography : Temporal databases. *ACM SIGMOD RECORD*Γ 15(4)Γ1986.

[MBJK90] J. MylopoulosΓA. BorgidaΓM. Jarke et M. Koubarakis. TELOS : representing knowledge about information systems. *ACM Transactions on Information Systems (TOIS)*Γ8(4):325–362Γ1990.

[MP92] Z. Manna et A. Pnueli. *The temporal logic of reactive and concurrent systems - Part. 1 Specification.* Springer VerlagΓ1992.

[MP93] A. Montanari et B. Pernici. Temporal reasoning. In Tansel et al. [TCG+93].

[NA87] S. B. Navathe et R. Ahmed. TSQL: a language interface for history databases. *Temporal Aspects in Information Systems*Γ1987.

[NA93] S. B. Navathe et R. Ahmed. Temporal extensions to the relational model and SQL. In Tansel et al. [TCG+93].

[OPS+95] T. OzsuΓR. PetersΓD. SzafronΓB. IraninΓA. Lipka et A. Munoz. TIGUKAT: A uniform behavioral objectbase management system. *The VLDB Journal*Γ 4:445–492Γ1995.

[Ora92] Oracle. *Oracle7 Server Administrator's Guide, Oracle part 6694-70-1292.* OracleΓ1992.

[PEA+95] J. PalazzoΓN. EdelweissΓE. ArrudaΓA. Laender et J Cavalcanti. Implementation of an object-oriented temporal model. Dans *proc. of the 6th International Conference on Database and Expert Systems and Applications (DEXA)*ΓLondon (UK)ΓSeptember 1995.

[Poi96] A. Poitou. *Un modèle d'événements duratifs.* Rapport de DEAΓUniversité Joseph Fourier – Institut National Polytechnique de GrenobleΓjuin 1996.

[Qui97] M.-C. Quidoz. Modèles et systèmes de bases de données temporelles et actives en économie. Mémoire CNAM en InformatiqueΓà soutenirΓdécembre 1997.

[Rod92] J. F. Roddick. Schema evolution in database systems : an annoted bibliography. *ACM SIGMOD RECORD*Γ21(4)ΓDecember 1992.

[RS93] E. Rose et A. Segev. TOOSQL a temporal object oriented query langage. Dans *proc. of the 12th International Conference on the Entity-Relationship approach, LNCS 823*ΓArlingtonΓTexasΓDecember 1993. Springer-Verlag.

[SA85] R. T. Snodgrass et I. Ahn. A taxonomy of time in databases. Dans *proc. of ACM SIGMOD*ΓMay 1985.

[Sar93] N. L. Sarda. HSQL : a Historical Query Language. In Tansel et al. [TCG+93].

[SC91] S. Su et H.-H. Chen. A temporal knowledge representation model OSAM/T and its query langage OQL/T. Dans *proc. of the 17th VLDB International Conference*ΓBarcelonaΓSeptember 1991.

[Sch95] S. Schwer. Structures temporelles pour les BD : état de l'art. Rapport interne du Laboratoire d'Informatique de Paris-NordΓ1995.

[Sci91] E. Sciore. Multidimensional versioning for object-oriented databases. Dans *proc. of the Second International Conference on Deductive and Object-Oriented Databases, LNCS 566*ΓMunichΓGermanyΓDecember 1991. Springer-Verlag.

[Sci94] E. Sciore. Versioning and configuration management in an object-oriented data model. *The VLDB Journal*Γ3:77–106Γ1994.

[SFC98] P.-C. SchollΓM.-C. Fauvet et J.-F. Canavaggio. Un modèle d'historique pour un SGBD temporel. *TSI, Numéro thématique "Bases de données"*Γ1998. À paraître.

[SFLM93] P.-C. SchollΓM.-C. FauvetΓF. Lagnier et F. Maraninchi. *Cours d'informatique - Langages et Programmation.* Collection Manuels Informatique. Masson (Paris)Γ Septembre 1993.

[SK94] S. Soukeras et P.-J.-H. King. TFDL: a temporal functional language for the management of historical databases. Dans *proc. of the 5th International Conference on Database and Expert Systems and Applications (DEXA), LNCS 856*Γ AthensΓGreeceΓ1994. Springer Verlag.

[SN97] A. Steiner et M.C. Norrie. Implementing temporal databases in object-oriented systems. Dans *proc. of the 5th International Conference on Databases for Advanced Applications*ΓMelbourneΓAustraliaΓApril 1997.

[Sno88] R. Snodgrass et al. Special issue on temporal databases. *Data Engineering Bulletin*Γ11(4)ΓDecember 1988.

[Sno92] R. T. Snodgrass. Temporal databases. Dans *proc. of the International Conference GIS - From space to territory*ΓPisa (Italy)ΓSeptember 1992. also in Theories and methods of Spatio-Temporal reasoning in Geographic SpaceΓ LNCS n°639ΓSpringer Verlag.

[Sno93] R. T. SnodgrassΓéditeur. *proc. of ARPA/NSF International Workshop on Infrastructure for Temporal Databases*ΓArlingtonΓTexasΓJune 1993.

[Sno94a] R. T. Snodgrass. Overwiew of the special session on temporal database infrastructure. *ACM SIGMOD RECORD*Γ23(1)ΓMarch 1994.

[Sno94b] R. Snodgrass et al. TSQL2 language specification. *ACM SIGMOD Record*Γ 23(1)ΓMarch 1994.

[Sno95a] R. T. Snodgrass. Temporal object-oriented databases: a critical comparison. Dans W. KimΓéditeurΓ*Modern database systems. The object model, interoperability and beyond*Γchapitre 19. Addison WesleyΓ1995.

[Sno95b] R. T. SnodgrassΓéditeur. *The TSQL2 temporal query language.* Kluwer Academic PublishersΓ1995.

[Soo91] M. D. Soo. Bibliography on temporal databases. *ACM SIGMOD Record*Γ 20(1):14–23ΓMarch 1991.

[SS92] M. Soo et R. Snodgrass. The TEMPIS project : mixed calendar query language support for temporal constants. TEMPIS Technical report n 29ΓDepartement of Computer ScienceΓUniversity of ArizonaΓTucsonΓmay 1992.

[SS93] A. Segev et A. Shoshani. A temporal data model based on time sequences. In Tansel et al. [TCG+93].

[SSD+92] M. SooΓR. SnodgrassΓC.E. DyresonΓC.S. Jensen et N. Kline. The TEMPIS project : Architectural Extensions to Support Multiple Calendars. TEMPIS Technical report n 32ΓDepartement of Computer ScienceΓUniversity of ArizonaΓ TucsonΓmay 1992.

[TAI87] *Proceedings of Conference on Temporal Aspects in Information Systems*ΓSophia-AntipolisΓMay 1987. AFCET.

[Tan93] A. U. Tansel. A generalized relational framework for modelling temporal data. In Tansel et al. [TCG+93].

[TCG+93] A. U. TanselΓJ. CliffordΓS. GadiaΓS. JajodiaΓA. Segev et R. SnodggrassΓ éditeurs. *Temporal Databases.* The Benjamins/Cummings Publishing CompanyΓ 1993.

[Tec95] O2 Technology. The O2 systemΓdocumentation v4.6ΓSeptember 1995.

[TIM95] Proceedings of time-95. Dans *TIME-95 International Workshop on Temporal Representation and Reasoning*ΓMelbourneΓAustraliaΓ1995.

[TSQ] TSQL2. représente toute une communauté de chercheurs qui ont travaillé à la définition d'un glossaire des concepts de base liés au tempsΓet des extensions qu'il faut apporter au modèle relationnel et à SQL pour représenter le temps dans une base de données. Les publications issues du groupe de travail sont accessibles via ftp anonyme à l'adresse cs.arizona.edu.

[Tur86] R. Turner. *Logiques pour l'Intelligence Artificielle.* MassonΓ1986.

[VGS96] C. VassilakisΓP. Georgiadis et A. Sotiropoulou. A comparative study of temporal dbms architecture. Dans *proc. of the 7th International Workshop on Database and Expert Systems and Applications (DEXA)*ΓZurich (Switzerland)ΓSeptember 1996.

[WD92] G.T.J. Wuu et U. Dayal. A uniform model for temporal object-oriented databases. Dans *proc. of the 8th IEEE Data Engineering Conference*Γ1992.

[WD93] G.T.J. Wuu et U. Dayal. A uniform model for temporal and versioned object-oriented databases. In Tansel et al. [TCG+93].

[WJ93] X. S. Wang et S. Jajodia. Temporal mediators as a way to support multiple temporal representations. In Snodgrass [Sno93].

[WJS93] X. S. WangΓS. Jajodia et V. S. Subrahmanian. Temporal modules : an approach toward federated temporal databases. Dans *proc. of the International Conference on Management of Data*. ACM SIGMODΓ1993.

[WJS95] X. S. WangΓS. Jajodia et V. S. Subrahmanian. Temporal modules : an approach toward federated temporal databases. *Information Systems*Γ82Γ1995.

[Yu 83] E. Yu Kandrashina. Representation of temporal knowledge. Dans *proc. of the 8th International Joint Conference in Artificial Intelligence*Γpages 346–358Γ Karlsruhe (DE)Γ1983.