Catégorie: jeunes chercheurs, application industrielle

TOOBIS: application de la gestion de Données Temporelles dans le domaine de la Recherche Clinique

Michael Souillard [1, 3] - Costas Vassilakis [2] - Anya Sotiropoulou [2]

(1) Centre Recherche Informatique
Université Paris I Sorbonne
90, rue Tolbiac
75013 Paris - France
Tél. : (33) 1-44-24-93-65
Fax : (33) 1-45-86-76-66

(2) Department of Informatics
University of Athens
Panepistimioupolis, TYPA Build.
15771 Athens - Greece
Tel : (30) 1-72-57-560
Fax : (30) 1-72-19-561

(3) Matra Systèmes & Informations
Parc d'affaire des portes
BP 613
27106 Val de Reuil - France
Tél. : (33) 2-32-63-40-00
Fax : (33) 2-32-63-42-00

Emails: {souillar@mcs-vdr.fr / mickael@univ-paris1.fr} {anya/costas}@di.uoa.gr

Résumé

Les données temporelles, c'est-à-dire des données évoluant dans le temps et pour lesquelles il est intéressant de maintenir un historique des valeurs, ne sont pas utilisées dans l'industrie. Mais cela ne veut pas dire pour autant que les diverses applications existantes ne gèrent que des données non temporelles. En fait des données temporelles doivent être gérées par un grand nombre d'applications et de secteurs industriels tels que la finance, l'assurance, le suivi médical, les services de réservation, etc. Ces applications utilisent le plus souvent le résultat de développement internes simulant des données temporelles, qui peuvent être coûteux. Le marché des solutions pour manipuler des données temporelles est encore limité, et ne répond pas à tous les besoins.
Cet article présente les résultats du projet européen TOOBIS - Temporal Object Oriented dataBase within Information System - en mettant l'accent sur une application gérant des données temporelles, développée dans le domaine de la Recherche Clinique. TOOBIS propose une extension du standard des bases de données orientées objet afin de fournir un système de gestion des bases de données temporelles complet, ainsi qu'une méthodologie de conception temporelle.

Mots clés

Temps, Objet, Base de Données, Modèle, Langage, Recherche Clinique

Abstract

Temporal data, i.e. data varying over time dimension whose history of evolutions are maintained, are not used in the industrial word. But, far from managing only non-temporal data, numerous and various applications and industrial sectors such as banking, insurance, disease management in medicine, booking and so on, face the management of temporal data. These applications often use results of own developments, simulating temporal data in a more or less effective ways.
This paper presents the results of the European project TOOBIS - Temporal Object Oriented dataBase within Information System - underlying an application using and managing temporal data, in the domain of Clinical Research. TOOBIS offers an extension of the object-oriented database standard in order to provide a full temporal object-oriented database management system, as well as a temporal methodology of analysis and design.

Keywords

Time, Object, Database, Model, Languages, Clinical Research

1. Introduction

Parmi les premiers travaux se penchant sur la dimension temporelle des informations, [Lang73] a introduit la notion de "e-fact", l'association d'un fait et d'un temps, comme la brique de base de la représentation de la connaissance chez l'Homme. La sémantique de cette estampille temporelle a été ensuite raffinée par [Bube77] en temps intrinsèque et temps extrinsèque. Le temps intrinsèque fait partie intégrante de l'information tandis que le temps extrinsèque représente un attachement dans un contexte temporel de cette information. Puis [Klop81] a introduit la définition d'un historique d'une entité - dans le modèle Entité-Relation - comme l'historique des attributs composant cette entité. Le temps ne contient plus une description du monde modélisé, mais ce temps est inclus dans la représentation de ce monde via des historiques. Quelques années plus tard, [Aria87] établit qu'afin de libérer la correspondance entre les états d'une base et le monde modélisé, il est nécessaire de manipuler plus d'une dimension temporelle. En fait depuis 20 ans, de nombreux travaux de recherche ont été effectués sur la gestion des données et des bases de données temporelles, comme en témoignent de nombreuses bibliographies sur ce sujet: [Mcke86], [StSn88], [Soo91], [Klin93], etc. Il en résulte un nombre important d'extensions des systèmes de bases de données. Pour le modèle Entité-Relation nous pouvons citer [Klop81] et [KlLo83]; de leur coté [Aria86], [Snod87], [TaCl93], s'attaquent au modèle Relationnel; et plus récemment le modèle Objet fait l'objet d'exploration via [RoSe93], [FaCa96], [StNo97]. Le principal résultat de ces années de recherche est [TSQL2], publié en 1995, qui définit une extension temporelle au standard relationnel SQL92. Ce travail, fruit d'une collaboration entre les principaux chercheurs impliqués dans le temps, apparaît comme un standard defacto des bases de données relationnelles temporelles.

Depuis les premiers travaux, la nomenclature a évolué, mais la sémantique reste très proche comme le montre les définitions des dimensions temporelles, définitions extraites de [JeCl93]:

- le **temps valide** d'un fait est le temps pendant lequel le fait est vrai dans le monde modélisé;

- le **temps transactionnel** d'un fait est le temps pendant lequel ce fait est présent dans la base de données, et peut être atteint;

- le temps utilisateur est un attribut non interprété, dont le domaine est une quantité de temps.

La prise en compte des deux premières dimensions temporelles conduit à une gestion bitemporelle des données.

De tous ces travaux théoriques sur la gestion des bases de données temporelles résultent un certain nombre d'implémentations reportées dans [Bohl95] - parmi lesquelles GCH-OSQL [CoPi95], Calenda [ScDi95], Chronolog [Bohl95b], TempCase [ThAi94], TempIS [AhSn86], TimeDB [TDB] and VT-SQL [VaLo95] - et plus récemment [StNo97b], TEMPOS [FaCa97], et TOOBIS qui est présentée dans cet article. 11 de ces implémentations ont eu lieu sur un SGBD relationnel, tandis que 5 ont été implémentées sur des SGBD orientés objet: GCH-OSQL, Calanda, [StNo97b], TEMPOS et TOOBIS[1] [TOOB97]. Calanda gère des séries temporelles évoluant sur le temps valide, utilisant des points temporels exprimés dans des calendriers multiples, et est implémenté sur le SGBD orienté objet ObjectStore. GCH-OSQL supporte le temps utilisateur et les objets évoluant sur le temps valide utilisant des points ou des intervalles de temps comme estampilles. Dans ce cadre, un modèle de données ainsi qu'un langage de requête introduisant deux nouvelles clauses dédiées à la manipulation des données temporelles sont

[1] TOOBIS - Temporal Object Oriented dataBase within Information System - projet européen Esprit IV #20671.

introduits et implémenté sur le SGBD orienté objet ONTOS. L'inconvénient de ces deux systèmes, sans prendre en compte les fonctionnalités offertes ou non, est qu'ils sont très liés aux SGBD sur lesquels ils sont implantés, rendant toute tentative de portage ardue. [StNo97] est implémenté sur O2, maintenant l'historique des objets évoluant sur le temps valide, mais il reste également très lié au système le supportant. Cependant, le langage de requêtes implémenté sur le langage OQL de O2 qui suit le standard ODMG - standard des bases de données orientées objet - devrait permettre un portage plus simple vers tout autre SGBD orienté objet également compatible avec le standard ODMG. En ce qui concerne TEMPOS, et plus précisément son langage TempOQL, celui-ci est une extension du langage de requête OQL, implémenté sur O2. TempOQL n'est pas un langage de manipulation de données temporelles, mais un langage de spécification et de manipulation de quantités temporelles, telles que les instants.

L'approche adoptée dans le cadre du projet TOOBIS, est de fournir une extension temporelle des SGBD orientés objet permettant la gestions des données temporelles, évoluant sur l'une ou les deux dimensions temporelles que sont les temps valide et transactionnel. Cette étude s'appuie sur une extension du standard ODMG [ODMG93], couvrant à la fois le modèle de données et les langages de définition et de requête. TOOBIS définit alors un SGBDOO-Temporel comprenant: un modèle de données temporel TODM (Temporal Object Data Model), un langage de définition temporel TODL (Temporal Object Definition Language) et un langage de requête temporel TOQL (Temporal Object Query Language). TOOBIS est implémenté au dessus du SGBD orienté objet O2, en C++, mais il se présente plus comme une validation de l'extension d'ODMG proposée dans ce projet, qu'une implémentation et une extension propre à O2.

Toujours dans le cadre du projet TOOBIS, une application pilote est réalisée dans le domaine de la Recherche Clinique, permettant de gérer les données provenant des essais cliniques. Le domaine médical est riche en terme de gestion de données temporelles, apparaissant comme application concrète de différents projets portant sur les bases de données temporelles, tels que [CoPi94] - [CoPi95] et [ThAi94].

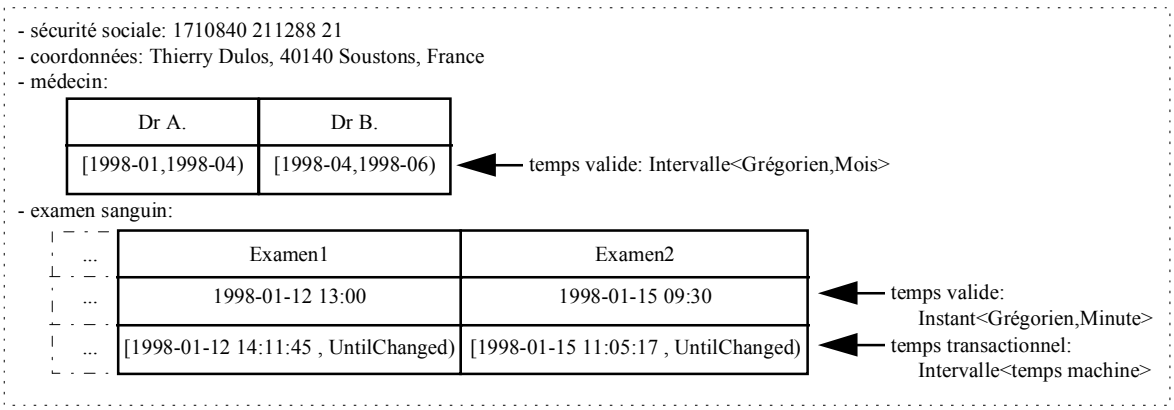
Pour poursuivre cet article, nous commencerons par donner quelques exemples simples de données temporelles dans un cadre médical. Puis une succincte présentation des différents composants de TOOBIS sera présentée. Des références pointeront vers d'autres documents plus techniques et plus longs pour compléter cette introduction. Dans la quatrième partie sera présentée l'application pilote développée dans le cadre du projet TOOBIS, pour la gestion des données résultantes d'essais cliniques. Enfin nous conclurons cette présentation, puis nous finirons par les annexes contenant la syntaxe de TOQL et une illustration de l'application gérant les données d'un essai clinique.

2. Illustration via quelques exemples

Illustrons les concepts de données temporelles via quelques exemples simples tirés du domaine médical. Prenons un patient hospitalisé. Ce patient possède un numéro de sécurité social qui est invariant, des coordonnées - nom, adresse - que l'on peut considérer comme invariant pour la durée de l'hospitalisation. Un patient est suivi par un médecin précis, mais ce médecin peut ne pas être le même pendant toute la durée du traitement. Cette information évoluant dans le temps, est donc une information temporelle, continue et variant sur l'axe de temps valide puisque l'on veut savoir à tout instant de l'hospitalisation, par quel médecin est suivi tel ou tel patient. Ce patient subit des examens médicaux, qui se répètent généralement un certain nombre de fois afin de suivre son évolution. Un de ces examens pourrait être une prise de sang, dont les observations mesurées peuvent être les taux de globules blancs et rouges, le taux de cholestérol, etc. Ces mesures

évoluent dans le temps, à chaque prise de sang, qui elle, est faite à un temps précis et connu, mais elles dépendent de cette prise de sang. Ces mesures sont donc des données temporelles discrètes, évoluant sur l'axe de temps valide. Si dans un souci de qualité et d'efficacité, on cherche également à connaître, par exemple, le délai entre une prise de sang et la disponibilité effective des résultats dans le système, il est alors intéressant de gérer également le temps transactionnel. Les résultats sanguins deviennent donc des données bitemporelles: le temps valide représente le temps du patient - le temps auquel la prise de sang a été effectuée - et le temps transactionnel représente le temps auquel les résultats ont été visibles et exploitables par le système.

La figure ci dessous, illustre cet exemple avec quelques valeurs qui seront réutilisées plus tard dans ce document. *UntilChanged* utilisé comme borne de fin des intervalles des estampilles du temps transactionnel, signifie que la valeur correspondante est toujours la valeur courante en base et qu'elle le restera jusqu'à ce qu'elle soit modifiée.



Exemple de données temporelles

3. Les composants de TOOBIS

Dans cette partie sont introduits les différents composants de TOOBIS: un modèle de données orienté objet et temporel, deux langages orientés objet et temporels de définition et de requête, ainsi qu'une méthodologie de conception orientée objet et temporelle.

3. 1 Modèle de données objet et temporel - TODM

3. 1.1 Modélisation et manipulation du temps

TODM reprend un modèle du temps standard, dont on peut retrouver les différents concepts et définitions dans le glossaire des bases de données temporelles [JeCl93] et dans [TSQL2]. Ce modèle est basé sur une structure linéaire et temporelle, sur laquelle un ordre total est défini via l'opérateur 'inférieurs à'. Une vue discrète de ce modèle est ensuite manipulée via un axe du temps. Les quantités temporelles suivantes sont définies:

- une période est une durée de temps de longueur connue, mais non ancrée sur l'axe du temps, comme par exemple '3 mois et 10 jours';

- un instant est un point sur l'axe du temps, comme par exemple, '1996-06-16 17:30:00';

- un intervalle est la quantité de temps comprises entre deux instants donnés de l'axe du temps, comme par exemple '[1971-06-16, 1996-06-16)';

- une chaîne d'intervalles est une union disjointe d'intervalles de l'axe du temps, comme par exemple '{ [1971, 1981), [1991, 1995) }'.

Chacune de ces quantités temporelles est exprimée dans un calendrier précis et avec une granularité, ou unité, précise. TODM implémente le calendrier Grégorien, ainsi que ses granules standards (année, mois, jour, heure, minute, seconde), supportant les années bissextiles et les secondes intercalaires, la spécification de fuseau horaire, etc. Mais TODM offre également un support multi-calendrier permettant la définition de calendriers-utilisateurs et de nouvelles granules - comme par exemple le pseudo calendrier des jours ouvrables - ainsi que la conversion d'une quantité temporelle exprimée dans un calendrier vers un autre calendrier.

Pour la manipulation des quantités temporelles, un ensemble d'opérations arithmétiques, d'opérateurs de comparaison et de changement d'unité, sont disponibles - ex: precedes, overlaps, merges, cast, etc. Elles sont définies et applicables sur les instants, les intervalles, les chaînes d'intervalles et les périodes. La manipulation d'instants et d'intervalles relatifs est également supportée, comme par exemple 't_1 se situe à t_0 plus 25 jours'.

3. 1.2 Rappels sur le modèle objet d'ODMG

Le modèle de données orienté objet de l'ODMG définit les caractéristiques des objets et comment ils peuvent être liés les uns aux autres. Le concept de base est celui de l'objet qui possède un identifiant unique invariant, appelé OID - Object IDentifier. L'état d'un objet est représenté par les valeurs de ses propriétés, attributs et relations, et son comportement est défini par un ensemble d'opérations applicables à tous les objets du même type.

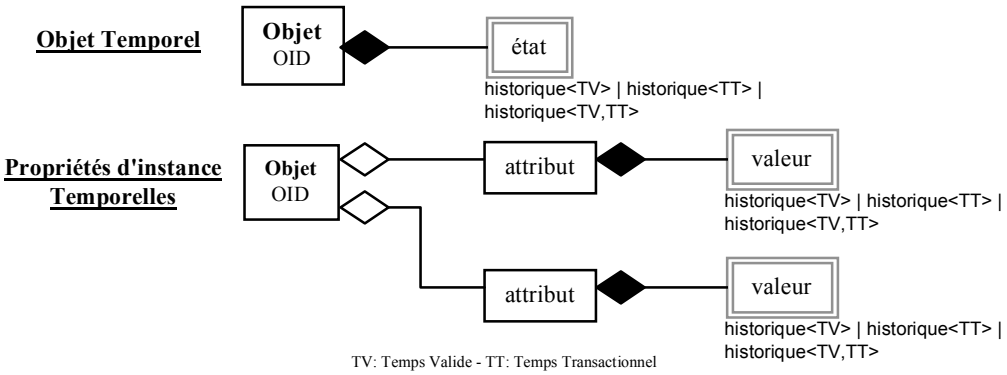
Un attribut est défini pour un type particulier, et prend ses valeurs dans l'ensemble des littéraux, qu'ils soient atomiques ou structurés. En ce qui concerne les relations, elles sont définies entre deux types, avec une cardinalité un-un, un-plusieurs ou plusieurs-plusieurs, ainsi que la possibilité de gérer un lien inverse assurant l'intégrité référentielle de la relation.

Sont également introduits dans ce modèle des littéraux et collections de base.

3. 1.3 Introduction du temps dans les concepts objet

Le but de TODM est d'introduire la gestion des données temporelles dans le monde des bases de données orientées objet via l'extension du modèle de données de l'ODMG. L'approche adoptée consiste à fournir les structures et les interfaces pour maintenir l'évolution des données variant sur l'axe du temps valide, du temps transactionnel ou sur ces deux dimensions temporelles, tout en respectant le modèle de base.

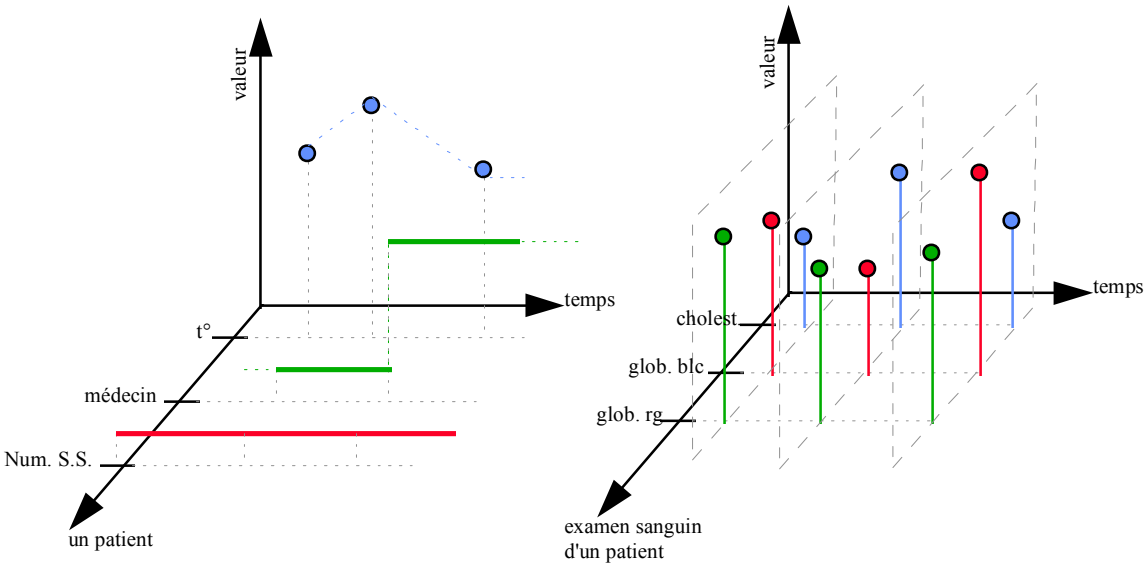
TODM introduit le temps au niveau de l'objet qui est le concept de base, mais également à un niveau plus fin que sont les propriétés d'instance - attributs et relations. En effet, un objet peut évoluer, sur l'une ou l'autre des dimensions temporelles, dans sa globalité; c'est à dire que son état évolue de manière globale: chacun des attributs et de ses relations évolue. Il est donc intéressant de pouvoir maintenir l'historique des évolutions de son état. D'un autre coté, un objet peut évoluer de manière partielle ou éparse: ses attributs et relations peuvent ne pas varier en même temps, peuvent varier sur des dimensions temporelles différentes, ou ne pas tous varier. Dans ce cas le degré de précision de l'historique doit être géré au niveau des propriétés d'instance et non plus au niveau de l'objet dans sa globalité. Ces deux cas s'excluent mutuellement. La figure suivante montre une modélisation de ces deux niveaux de caractéristiques temporelles, objet temporel et propriété d'instance temporelle, en utilisant la symbologie de la méthodologie TOOM, introduite dans la partie 3.4 de ce document.



Objet temporel vs propriété d'instance temporelle

TODM représente ces concepts via les types Temporal_Object, Temporal_Atttribute et Temporal_Relationship, sous-types des types Object, Attribute et Relationship définis dans le modèle d'ODMG.

En reprenant l'exemple décrit précédemment, les caractéristiques temporelles au niveau de l'objet peuvent être illustrées par l'examen sanguin dont les mesures évoluent simultanément à chaque nouvelle prise de sang. Tandis que pour représenter le patient qui contient des informations invariantes et temporelles, il est plus intéressant et commode d'utiliser des attributs et relations temporels.



Exemple de propriétés d'instance temporelles vs un objet temporel

Les différents concepts de TODM sont abordés plus en détail dans [MS&I97] et [SoSo98].

3. 2 Langage de définition objet et temporel - TODL

3. 2.1 Rappels sur le langage de définition d'ODMG

Le langage de définition objet (ODL), proposé par ODMG, est une extension du langage IDL proposé par OMG pour CORBA. ODL permet au programmeur de définir les interfaces des objets persistants, sous la forme de types appelés *interfaces*. Avec ODL on peut définir les caractéristiques suivantes pour chaque interface: ses super-types, son extension, ses clés, ses

propriétés d'instance - attributs et relations, les signatures des opérations qui peuvent être appliquées à ses instances.

ODL n'est pas un langage de manipulation, par conséquent, on ne peut pas définir les corps des opérations. Ceci est fait en utilisant des langages de manipulation, comme C++ ou Java.

3. 2.2 Définition de quantités et caractéristiques temporelles via TODL

TODL étend le langage de définition d'ODMG-93, complétant ainsi les définitions d'interface disponibles dans ODL par la possibilité de définir des caractéristiques temporelles. TODL est en fait un langage de définition simple permettant de mettre à disposition de l'utilisateur toutes les structures temporelles plus ou moins complexes fournies par TODM.

Le temps utilisateur est supporté via la possibilité d'utiliser les types Instant, Interval et Period, comme tout autre littéral de base. Pour ces types il est également possible de définir le calendrier et la granularité, et de préciser s'il s'agit d'un temps absolu ou relatif.

TODL permet la définition d'interfaces de données temporelles évoluant sur l'une ou les deux dimensions temporelles - valide ou transactionnelle - au niveau de l'objet ou au niveau des propriétés d'instance. Pour définir le temps valide on utilise le mot clé *valid* et pour définir le temps transactionnel on utilise le mot clé *transaction*. Pour le temps valide, le programmeur peut également définir le calendrier et la granule dans lesquels seront exprimées les estampilles de temps valide. Pour le temps transactionnel, ces informations sont imposées automatiquement par le système.

TODL étend également la définition des clés de ODL pour permettre la définition de clés temporelles, comme par exemple la clé *point* (valeur par défaut) qui, définie sur un attribut temporel, signifie que pour une même instance, cet attribut ne peut avoir deux valeurs différentes en un même instant.

En reprenant l'exemple illustré précédemment, les définitions de ces entités temporelles, via TODL pourraient être:

```
interface Patient
( key numSS; )
{   attribute string numSS;
    attribute string medecin valid state granularity month;
    attribute float temperature valid event granularity minute;
}
interface ExamenSanguin valid event granularity minute transaction
( )
{   attribute float cholest;
    attribute float globBlanc;
    attribute float globRouge;
}
```

La syntaxe est abordée plus en détail dans [UoA97a], [UoA97c] et [SoSo98].

3. 3 Langage de requête objet et temporel - TOQL

TOQL est une extension du langage de requête objet OQL de ODMG-93, version 2.0, qui permet la gestion de requêtes sur des données temporelles et non temporelles d'une manière uniforme.

La description de la syntaxe du langage TOQL est présentée en Annexe 1 de cet article.

3. 3.1 Rappels sur le langage de requête d'ODMG

Le langage de requête de ODMG (OQL) permet à l'utilisateur de sélectionner les données présentes en base. La version 2.0 de OQL est très proche de SQL et est déjà présente dans les SGBD orientés objet comme O2, Poet.

La distinction principale entre OQL et SQL est le fait que le premier n'est qu'un langage de requête, et qu'il faut avoir recourt aux langages de manipulation tels que C++ ou Java, pour introduire et modifier des données dans la base.

3. 3.2 Requêtes sur des données temporelles via TOQL

L'un des problèmes des langages de requêtes temporels est la gestion des requêtes non temporelles - c'est-à-dire les requêtes conventionnelles: elles doivent produire le même résultat que dans le langage de requêtes original, pour des raisons de compatibilité ascendante évidente. Les requêtes OQL, qui sont des requêtes TOQL valides, retourneront toujours - même quand elles sont appliquées sur des données temporelles - la version courante de la donnée: la valeur en base pour une donnée non temporelle, et la valeur valide au moment de l'exécution de la requête pour une donnée temporelle. Par exemple, la requête suivante retournant le médecin du patient dont le numéro de sécurité sociale est '1710840 211288 21', effectuée sur les données de l'exemple de la partie 2, à la date du 15 février 1998, retournera le 'Dr A.':

```
select p→medecin
from Patients as p
where p→numSS = '1710840 211288 21'
⇒
'Dr A.'
```

Il est possible de sélectionner l'historique entier d'une donnée temporelle en utilisant le mot clé *valid* pour obtenir l'historique complet sur la dimension valide d'une donnée valide ou bitemporelle, le mot clé *transaction* pour obtenir l'historique complet sur la dimension transactionnelle d'une donnée transactionnelle ou bitemporelle, et le mot clé *bitemporal* pour obtenir l'historique complet sur les deux dimensions temporelles d'une donnée bitemporelle. Toujours avec le même exemple, la requête suivante permet de récupérer l'historique complet des examens sanguins du patient dont le numéro de sécurité sociale est '1710840 211288 21':

```
select bitemporal p→examen_sanguin
from Patients as p
where p→numSS = '1710840 211288 21'
⇒
{ ( Examen1, '1998-01-12 13:00', '[1998-01-12 14:11:45,UntilChanged)' ),
  ( Examen2, '1998-01-15 09:30', '[1998-01-15 11:05:17,UntilChanged)' ) }
```

TOQL permet également d'utiliser les données temporelles comme des listes, accessible via un index. Cet index peut être un ou deux nombres entiers spécifiant la ou les positions des variants à retourner. Il peut aussi être un instant, retournant alors les variants dont l'estampille temporelle contient cet instant. Et enfin, cet index peut être composé d'un intervalle, ou de deux instants représentant les bornes d'un intervalle, les variants retournés seront alors les variants dont l'estampille temporelle chevauche cet intervalle. Les expressions *valid at* et *current at* permettent de préciser que la sélection s'applique sur le temps valide et/ou le temps transactionnel, lorsque cet index est appliqué à une entité bitemporelle. La requête suivante liste les médecins qui ont été en charge du patient dont le numéro de sécurité sociale est '1710840 211288 21', entre le 1er février et le 1er mai 1998, ainsi que l'intervalle correspondant:

```
select (valid p→medecin)[period '[1998-02,1998-05)' granularity day]
from Patients as p
where p→numSS = '1710840 211288 21'
⇒
```

```
{ ( 'Dr A.', '[1998-02-01,1998-04-01)' ),
  ( 'Dr B.', '[1998-04-01,1998-05-01)' ) }
```

De manière générale, on peut utiliser des données temporelles où on peut utiliser des collections, comme par exemple: pour définir une variable de la clause from, pour définir une variable de quantification existentielle, pour définir une variable de quantification universelle, etc.

Ceci n'illustre qu'une infime partie des fonctionnalités offertes par TOQL. Il faut y ajouter la possibilité de convertir des objets temporels en collections de valeurs, et inversement, un opérateur de jointure *tstruct*, des opérateurs de restructuration des estampilles temporelles pour produire par exemple des intervalles maximum, le partitionnement d'une entité temporelle, l'agrégation de plusieurs entités temporelles, etc. Pour plus d'information, se reporter à [UoA97b], [UoA97d] et [SoSo98].

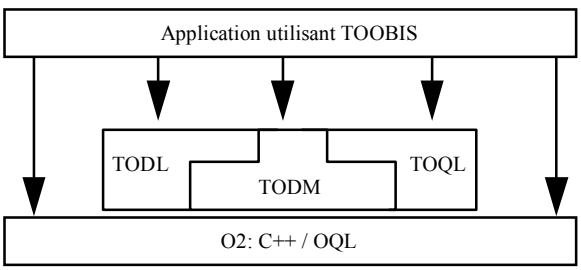
3. 4 Méthodologie orientée objet et temporelle - TOOM

L'introduction d'aspects temporels dans la conception d'applications utilisant des bases de données, nous amène à nous poser des questions telles que "Quelles sont les unités qui rythment les données de l'application?", "Quelles sont les données évoluant dans le temps dont il est pertinent de maintenir un historique?", etc. TOOM répond à ces questions en fournissant une extension des paradigmes des méthodologies orientées objet, en introduisant la mesure et la manipulation du temps, ainsi que la gestion d'historiques des données. Une caractéristique importante de TOOM est sa compatibilité avec UML, standard de base des méthodologies d'analyse et de conception orientées objet.

En résumé, TOOM a pour but d'aider à décrire les aspects temporels d'une application utilisant des bases de données, et de supporter le mapping des spécifications conceptuelles vers les structures de TODM via des déclarations TODL. cf. [Sorb97].

3. 5 Implémentation et utilisation

Les différents composants de TOOBIS sont implémentés en tant que librairies C++ au dessus du SGBD orienté objet O2, sur Solaris OS. Ils sont également portés sur Windows NT 4.0.



Les composants de TOOBIS implémentés O2

3. 5.1 TODM

TODM se présente sous la forme d'une librairie de classes C++ utilisable dans tout programme, C++ sur O2.

3. 5.2 TODL

TODL est un exécutable prenant en entrée un fichier contenant des définitions d'interfaces TODL, et dont le résultat est la génération des fichiers C++ (.hxx et .cc) contenant les définitions des

classes C++ correspondantes. Ces définitions seront importées dans O2 par le programmeur, qui doit également compléter le code des méthodes définies dans ces classes.

3. 5.3 TOQL

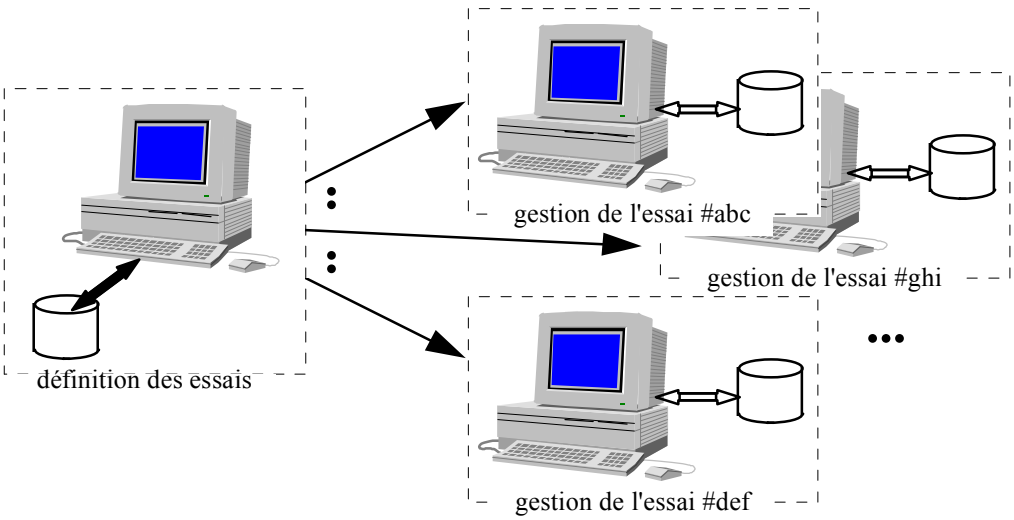
Le processeur TOQL est implémenté comme une librairie C++ fonctionnant au dessus du processeur OQL de O2. Les requêtes TOQL sont transformées en requêtes OQL, utilisant les méthodes des classes de TODM, puis elles sont passées au processeur OQL de O2. Les résultats du processeur OQL sont retournés à l'utilisateur ou à l'application qui a formulé la requête.

On peut utiliser TOQL soit de manière interactive - requêtes formées ad hoc - soit via un programme C++ de manière similaire au lancement de requête OQL.

4. Application à la Recherche Clinique: gestion des données provenant d'essais cliniques

Dans le cadre du projet TOOBIS, une application pilote utilisant les différents composants TOOBIS est développée dans le domaine de la Recherche Clinique, et plus précisément pour la gestion des données résultantes d'Essais Cliniques effectués sur des patients. Un essai clinique consiste à tester l'efficacité et la tolérance de tel ou tel traitement sur un ensemble de patients, et sur une durée plus ou moins longue.

Ce pilote se découpe en deux niveaux: d'un coté une application permettant de définir les caractéristiques des essais cliniques et de l'autre coté les applications générées par la précédente, gérant les données résultantes d'un essai clinique donné. Le schéma suivant illustre ces deux niveaux d'applications, qui seront détaillés dans les prochains paragraphes.



Deux niveaux d'applications: définition des essais, gestion d'un essai donné

Les données résultantes d'un essai clinique sont en grande partie des résultats de tests et d'observations effectués sur les différents patients inclus dans cet essai. Ces données sont de toute évidence des données temporelles puisqu'elles représentent l'évolution du patient sur la durée de l'essai: ces données évoluent sur le temps valide.

De plus, une des fonctionnalités principales de la gestion de ces données est la clarification ou la détection d'erreurs et d'incohérences, afin que l'analyse biostatistique qui en sera faite plus tard ne soit pas entachée de données erronées. Lors de cette phase de clarification, de nombreuses corrections sont effectuées sur les données, et dans un objectif orienté audit de qualité, il est

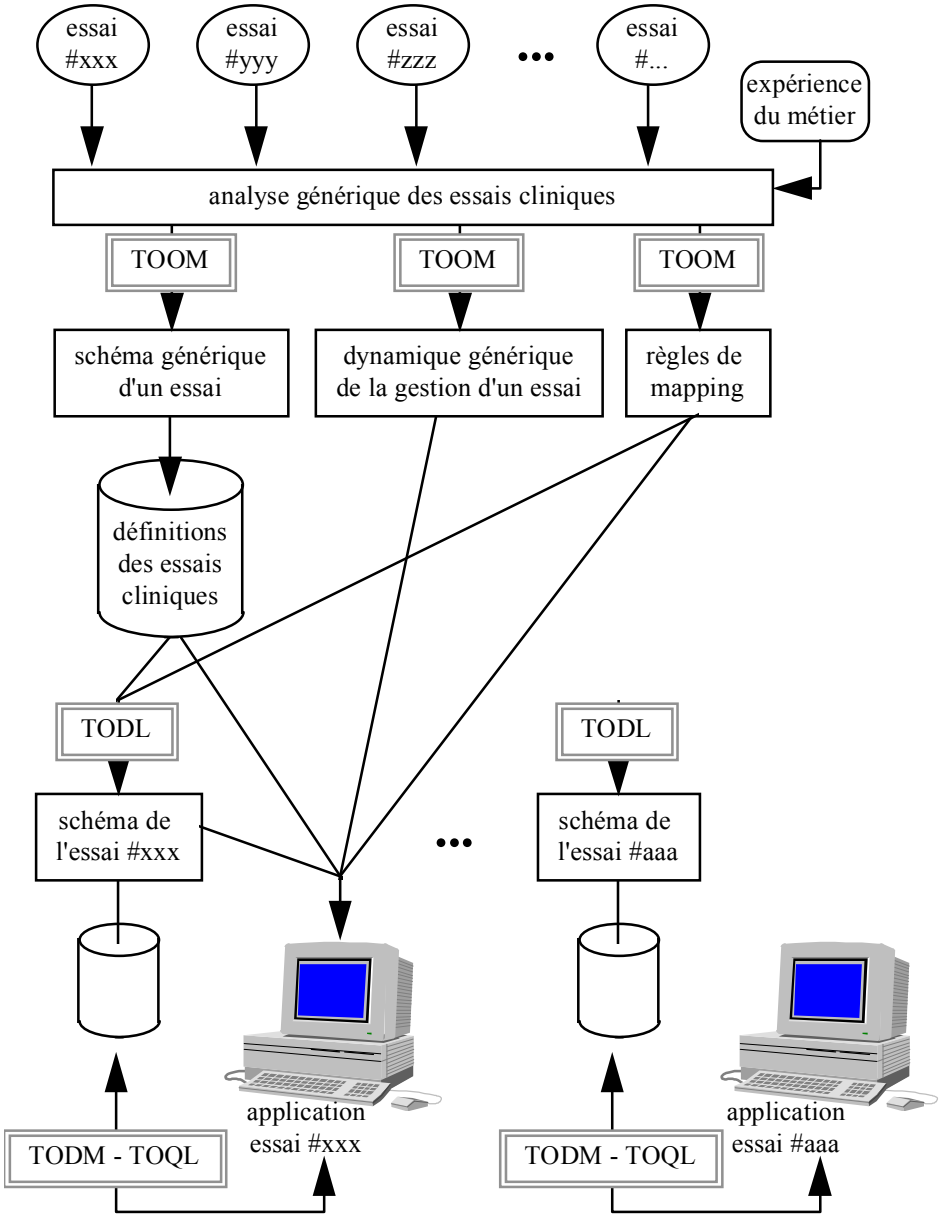
nécessaire de garder trace de toutes les modifications (qui, quoi, quand) afin de tenter d'améliorer la mise en œuvre des prochains essais cliniques. Ces données évoluent donc également dans le temps machine, sur l'axe transactionnel, maintenant ainsi l'audit complet des données présentes en base.

Par conséquent, une grande majorité des données provenant des essais cliniques sont des données bitemporelles.

4. 1 Définitions des Essais Cliniques

Chaque essai clinique est différent, portant sur tel ou tel traitement, contre telle ou telle maladie. Donc pour gérer et en particulier saisir les données patients, chaque essai possède une application qui lui est propre. Cependant tous les essais se ressemblent d'un point de vue mise en œuvre, organisation et collecte des informations. Les données sont recueillies sous la forme de cahiers, ces cahiers sont découpés en visites, chaque visite contient des série de tests, chaque test regroupe des observations, etc. Cette architecture précise a conduit à adopter une approche générique, valide pour tous les essais. Cette approche repose dans la définition générique d'un essai clinique, tant au point de vue des structures de données qu'au point de vue dynamique de l'application qui gérera ces données. Un schéma de données permettant de définir tout essai clinique a été élaboré. Ce schéma est instancié à chaque définition d'un nouvel essai. Grâce à des règles de mapping qui ont été définies dans la phase d'analyse générique des essais cliniques, les données définissant un essai sont alors utilisées pour créer le schéma de données propre à cet essai, ainsi que pour générer l'application de gestion des données provenant de cet essai.

Le schéma ci-dessous, illustre cette démarche générique.

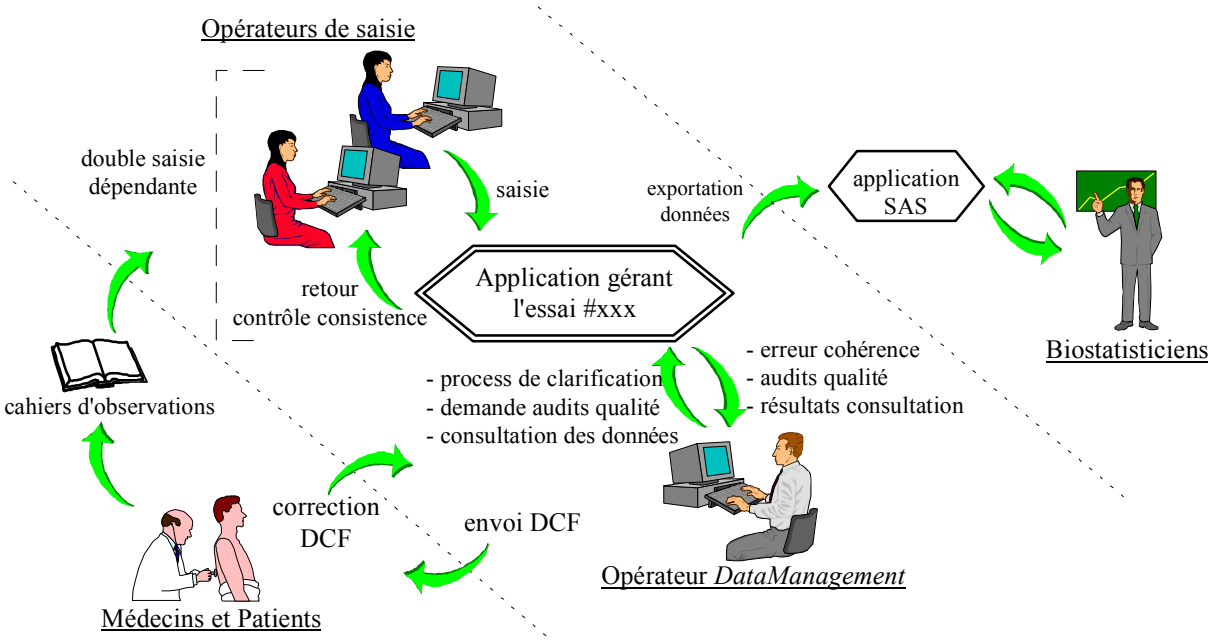


Approche générique permettant la définition des essais et la génération de leur application

Cette figure montre également l'emploi des différents composants de TOOBIS. La méthodologie TOOM a été utilisée pour modéliser la représentation générique d'un essai clinique. Ses composantes temporelles ont permis de modéliser la dynamique de la gestion des données temporelles d'un essai, ainsi que la construction des règles de génération pour les schémas de données temporelles des essais et pour leur application de gestion. Le schéma générique d'un essai est un schéma de données classique, i.e. sans caractéristique temporelle. Tandis que le schéma de donnée propre à un essai modélise des données temporelles. Pour créer ce schéma temporel, des déclarations d'interfaces TODL sont générées à partir des données décrivant la structure de cet essai et des règles de mapping. Le schéma de données temporelles ainsi obtenu est alors utilisé dans l'application de gestion correspondante. Enfin les liaisons entre les applications et les bases de données où sont stockées les données temporelles sont effectuées en utilisant les méthodes des classes de la librairie TODM qui permettent d'insérer, de modifier et d'atteindre les données, ainsi que via le langage de requête TOQL utilisé par exemple pour les audits de qualité, les contrôles de cohérences, etc.

4. 2 Gestion des données d'un essai clinique donné

Une première approche descriptive de l'application gérant les données résultantes d'un essai clinique est présentée dans le schéma suivant.



Gestion des données d'un essai clinique

Tout d'abord les données recueillies par les médecins sont stockées en base par le biais d'une double saisie dépendante: une première saisie est effectuée, puis une deuxième au cours de laquelle les éventuelles inconsistances par rapport aux premières valeurs saisies doivent être levées par le second opérateur. Il peut donc déjà y avoir un historique avec deux valeurs sur la dimension transactionnelle pour une même donnée en base.

Puis le processus de clarification est lancé sur les données présentes et courantes en base. Des contrôles de cohérence et de détection d'erreurs sont effectués. Les éventuelles incohérences, due à des erreurs de saisie, des erreurs d'observations, des valeurs manquantes, etc., sont retournées sous la forme de formulaires de clarification de données - DCF. Les DCF sont ensuite envoyées aux médecins ayant suivis les patients sur lesquels les incohérences ont été détectées, qui corrigent les valeurs erronées ou complètent les valeurs manquantes. Ces corrections sont ensuite apportées en base.

Le processus de clarification se poursuit jusqu'à l'obtention d'une base *propre*, où aucune incohérence n'est alors détectée. Les données propres peuvent alors être exportées vers l'application SAS qu'utilisent les biostatisticiens pour évaluer l'essai clinique suivant des critères valorisant ou non tel ou tel traitement testé lors de cet essai.

Le processus complet de gestion des données fait clairement apparaître l'évolution des données sur l'axe du temps transactionnel, i.e. leur évolution dans le temps machine, puisque les données sont contrôlées et corrigées jusqu'à l'obtention d'une base propre. Cependant le temps valide est également bien présent puisque les mêmes observations apparaissent à des visites différentes et par conséquent les même interfaces de saisie peuvent être utilisées. En Annexe 2 est présentée une copie d'écran, extraite de l'application d'un essai dédié au traitement de l'asthme. Cette image

montre une phase de saisie de données. Mais pour des raisons de confidentialité, les données présentes sont fictives et n'illustrent que des informations standards.

5. Conclusion

Cet article présente les résultats du projet européen TOOBIS, visant à étendre la gestion des données temporelles au monde des systèmes d'information orientés objet. Un SGBD Orienté Objet et Temporel comprenant un modèle de données, un langage de définition et un langage de requête ont été spécifiés puis implémentés au dessus d'un SGBDOO existant, O2. Une méthodologie de conception orientée objet et temporelle a également été produite. Une validation des ces résultats via une application pilote dans le domaine de la Recherche Clinique a permis de montrer le côté prometteur de cette étude. Cette application gère la définition d'essais cliniques et la génération d'applications propre à la gestion des données de chaque essai.

Le caractère porteur de notre approche se situe dans l'intégration de deux domaines, celui des bases de données temporelles et celui de l'approche objet, tout en s'appuyant sur de réels besoins utilisateurs. Notre approche s'inscrit dans une optique *formalisation driven*, s'appuyant sur les standards tels que ODMG-93 pour les bases de données objet, et UML pour les méthodologies d'analyse et de conception. La gestion des données résultantes d'essais cliniques n'est qu'un exemple d'utilisation du SGBDOO-T TOOBIS. Il trouverait également sa place dans d'autres secteurs d'activités, comme c'est le cas actuellement avec une autre application pilote dédiée à l'optimisation de production et de transport de denrées périssables.

L'orientation industrielle de cette étude a conduit à écarter certains points, comme par exemple la gestion des historiques des évolutions de schéma de données dans un SGBD. Enfin, il reste à mener une étape d'industrialisation visant à transformer TOOBIS en un véritable produit logiciel.

6. Références

[AhSn86] I. Ahn, R.T. Snodgrass. Performance Evaluation of a Temporal Database Management System. Conférence Internationale Management of Data, 1986.

[Aria86] G. Ariav. A Temporally Oriented Data Model. ACM Database Systems, Vol.11, No 4, 1986.

[Aria87] G. Ariav. Design Requirements for Temporally Oriented Information Systems. Temporal Aspects in Information Systems (TAIS), Sophia Antipolis, France, 1987 (p. 3-16).

[Bohl95] M. Bohlen. Temporal Databases System Implementations. Département de Mathématiques et d'Informatique, Université d'Aalborg, 1995.

[Bohl95b] M. Bohlen. Chronolog 4.0 Reference Manual. Institute for Electronic Systems, Université d'Aalborg, 1995.

[Bube77] J.A. Bubenko. The Temporal Dimension in Information Modeling. Architecture & models in Data Base Management Systems. G.M. Nijssen (Ed.), North Holland Publishing Co., Amsterdam 1977 (p. 93-118).

[CoPi94] C. Combi, F. Pincrioli et al. Managing & displaying different time granularities of clinical information. 18ème Symposium on Computer Applications in Medical Care, Washington, 1994.

[CoPi95] C. Combi, F. Pincrioli et al. Querying Temporal Clinical Database with Different Time Granularities: the GCH-OSQL Language. 19ème Symposium on Computer Applications in Medical Care, Philadelphia, 1995.

[FaCa96] M.C. Fauvet, J.F. Canavaggio, P.C. Scholl. Expressions de requêtes temporelles dans un SGBD à objets. 12ème Journées Bases de Données Avancées, BDA'96, 1996.

[FaCa97] M.C. Fauvet, J.F. Canavaggio, P.C. Scholl. Manipulation de valeurs temporelles dans un SGBD à objets. 15ème Congrès Inforsid, Toulouse, France, 1997.

[GaRo95] S. Gamerman, C. Rolland, C. Souveyet, A. Benjamen. Aspects temporels et historiques dans les bases de données des S.I.C.: état de l'art. Etude #9473326 Matra Cap Systèmes, 1995.

[JeCl93] C.S. Jensen, J. Clifford, R. Elmarsi et al. A Consensus Glossary of Temporal Database Concepts. 1993.

[Klin93] Klinc. ACM SIGMOD Vol.22, No 1, 1993.

[Klop81] M.P. Klopprogge. TERM: an approach to include time dimension in the Entity-Relationship Model. Entity-RelationshipApproach to Information Modeling & Analysis. P.P.S. Chen (Ed.), ER Institute, 1981 (p. 477-512).

[KlLo83] M.P. Klopprogge, P.C. Lockeman. Modeling informations preserving information databases: consequences of the concept of time. 9ème Conférence VLDB, Florence, Italie, 1983.

[Lang73] B. Langefors. Theoretical Analysis of Information Systems. Student Literature [Lund, Sweden], 1973 (2ème édition).

[Mcke86] E. Mckensie. Bibliograhy: Temporal Databases. ACM SIGMOD Vol.15, No 4, 1986.

[MS&I97] Matra Systèmes & Information, O2 Technology. TODM Specifications & Design. Rapport du projet TOOBIS. http://www.di.uoa.gr/~toobis

[ODMG93] R.G.G. Catell, T. Atwood et al. ODMG-93. International Thomson Publishing, 1994.

[RoSe93] E. Rose, A. Segev. A Temporal Object-Oriented Query Language. Lectures Notes in Computer Science No 823, 1993.

[ScDi95] D. Schmidt, A.K. Dittrich et al. Time series, a Neglected Issue in Temporal Database Research?, International Workshop on Temporal Databases, Zurich, 1995.

[Snod87] R.T. Snodgrass. The Temporal Query Language TQuel. ACM Database Systems, Vol.12, No 2, 1987.

[Soo91] M. Soo. Bibliography on temporal databases. Conférence ACM SIGMOD, 1991.

[Sorb97] Université Paris I Sorbonne. Temporal Object Oriented Methodology. Rapport du projet TOOBIS. http://www.di.uoa.gr~toobis

[SoSo98] Anya Sotiropoulou, Michael Souillard, Costas Vassilakis. Temporal Extension To ODMG. Issues & Applications of Database Technology, Berlin, 1998. (à paraître)

[StSn88] R. Stam, R.T. Snodgrass. A Bibliography of Temporal databases. Database Engineering, Vol.7, No 4, 1988.

[StNo97] A. Steiner, M.C. Norrie. A Temporal Extension to a Generic Object Data Model. 9ème CAiSE, Rapport Technique TimeCenter No 15, 1997.

[StNo97b] A. Steiner, M.C. Norrie. Implementing Temporal Databases Object-Oriented Systems. 5ème DASFAA, Rapport Technique TimeCenter No 14, 1997.

[TaCl93] A.U. Tansel, J. Clifford et al. Temporal Databases: Theory, Design and Implementation. Benjamin/Cummings, 1993.

[TDB] TimeCenter. Notes and README files from TimeDB Software.

[ThAi94] B. Theodoulidis, A. Ait-Braham, G. Karvelis. The ORES Temporal DMBS and the ERT-SQL Query Language. 5Ôme Conférence Database and Expert System Applications, Athènes, 1994.

[TOOB97] http://www.di.uoa.gr/~toobis

http://panoramix.univ-paris1.fr/CRINFO/ANI/toobis.html

[TSQL2] TSQL2 Language Design Committee, R.T. Snodgrass. TSQL2. Kluwer Academic Publishers, 1995.

[UoA97a] Université d'Athènes, 01 Pliroforiki, O2 Technology. TODL Specification and Design", Rapport du projet TOOBIS. http://www.di.uoa.gr/~toobis

[UoA97b] Université d'Athènes, 01 Pliroforiki, O2 Technology. TOQL Specification and Design", Rapport du projet TOOBIS. http://www.di.uoa.gr/~toobis

[UoA97c] Université d'Athènes, 01 Pliroforiki, O2 Technology. TODL Manual", Rapport du projet TOOBIS. http://www.di.uoa.gr/~toobis

[UoA97d] Université d'Athènes, 01 Pliroforiki, O2 Technology. TOQL Manual", Rapport du projet TOOBIS. http://www.di.uoa.gr/~toobis

[VaLo95] C. Vassilakis, N. Lorentzos, P. Georgiadis. Transaction Support in a Temporal DBMS. International Workshop on Temporal Databases, Zurich, 1995.

7. Annexes

7. 1 Annexe 1: Syntaxe BNF de TOQL

Les notations suivantes sont utilisées:

- {symbol} signifie une séquence de 0 ou n symboles;
- [symbol] signifie un symbole optionnel;
- **keyword** est un terminal de la grammaire;
- nom_xxx représente un identifiant;
- xxx_literal se suffit à lui-même, par exemple integer_literal représente un entier;
- bind_argument est un paramètre pour le binding dans un langage de programmation.

7. 1.1 Axiom

```
query_program ::= {define_query;} query
define_query ::= define identifier as
  query
```

7. 1.2 Basic

```
query ::= nil
query ::= true
query ::= false
query ::= integer_literal
query ::= float_literal
query ::= character_literal
query ::= string_literal
query ::= instant_literal
query ::= interval_literal
query ::= period_literal
query ::= entry_name
query ::= query_name
query ::= bind_argument
query ::= from_variable_name
query ::= (query)
```

7. 1.3 Simple Expression

```
query ::= query + query
query ::= query - query
query ::= query * query
query ::= query / query
query ::= - query
query ::= query mod query
query ::= abs (query)
query ::= query || query
```

7. 1.4 Comparison

```
query ::= query comparison_operator query
query ::= query like string_literal
comparison_operator ::= =
comparison_operator ::= !=
comparison_operator ::= <
comparison_operator ::= <=
comparison_operator ::= >
comparison_operator ::= >=
```

7. 1.5 Boolean expression

```
query ::= not query
query ::= query and query
query ::= query or query
operator && is synonymous with and
operator || is synonymous with or
operator ! is synonymous with not
```

7. 1.6 Constructor

```
query ::= type_name ([query])
query ::= type_name (identifier: query {,
  identifier: query})
query ::= struct(identifier: query {,
  identifier: query})
query ::= set([query {, query}])
query ::= bag([query {, query}])
query ::= list([query {, query}])
query ::= array([query {, query}])
query ::= period_set([query {, query}])
```

7. 1.7 Accessor

```
query ::= query dot attribute_name
query ::= query dot relationship_name
query ::= query dot operation_name([query
  {, query}])
dot ::= . | ->
query ::= query [query]
query ::= query [query : query]
query ::= query [distinct query]
query ::= query [valid at [distinct]
  query]
query ::= query [current at query]
query ::= query [valid at [distinct]
  query, current at query]
query ::= query [current at query, valid
  at [distinct] query]
query ::= first(query)
query ::= last(query)
query ::= function_name(query {, query})
```

7. 1.8 History accessor

```
query ::= valid query
query ::= transaction query
query ::= bitemporal query
```

7. 1.9 Timestamp Accessor

```
query ::= valid(query)
query ::= transaction(query)
```

7. 1.10 Collection expression

```
query ::= for all identifier in query:
  query
query ::= exists identifier in query:
  query
query ::= exists (query)
query ::= unique (query)
query ::= query in query
query ::= query comparison_operator
  quantifier query
quantifier ::= some
quantifier ::= any
quantifier ::= all
query ::= count(query)
query ::= count(*)
query ::= sum(query)
query ::= min(query)
query ::= max(query)
query ::= avg(query)
```

7. 1.11 Select Expression

```
query ::= select [distinct]
  projection_attributes
   from variable_declaration
       {, variable_declaration}
   [where query]
   [group by partition_attributes]
   [having query]
   [order by sort_criterion {,
       sort_criterion}
projection_attributes ::= projection { ,
  projection}
projection_attributes ::= *
projection ::= query [as identifier]
variable_declaration ::= query [[as]
  identifier]
partition_attributes ::= projection { ,
  projection}
partition_attributes ::= time_axis query
  [leading query] [trailing query]
time_axis ::= valid
time_axis ::= transaction
sort_criterion ::= query [ordering]
ordering ::= asc
ordering ::= desc
```

7. 1.12 Set expression

```
query ::= query intersect query
query ::= query union query
query ::= query except query
operator + is synonymous with union
operator - is synonymous with except
operator * is synonymous with intersect
```

7. 1.13 Conversion

```
query ::= listtoset(query)
query ::= element(query)
query ::= flatten(query)
query ::= (class_name)query
query ::= valid (state | event) [overlap]
  [granularity] [calendar calendar_spec]
  [query : query]
query ::= transaction [query]
query ::= bitemporal (state | event)
  [overlap] [granularity] [calendar
  calendar_spec] [query : query]
```

7. 1.14 Temporal Modifiers

```
query ::= valid [state] query
query ::= transaction [state] query
query ::= bitemporal [state] query
```

7. 1.15 Temporal join

```
query ::= tstruct(identifier: query {,
  identifier: query})
```

7. 1.16 Restructuring operator

```
query ::= query (partition time_axis as
  restruct_spec)
restruct_spec ::= instant
restruct_spec ::= period
```

7. 2 Annexe 2 : Exemple d'interface de saisie de données d'un essai clinique