Un modèle de données pour les requêtes sensibles au contexte dans les environnements "indoor"

Imad Afyouni, Cyril Ray, Sergio Ilarri, Christophe Claramunt

Un modèle de données pour les requêtes sensibles au contexte dans les environnements “indoor”

Imad Afyouni[1], Cyril Ray[1], Sergio Ilarri[2] and Christophe Claramunt[1]
[1]Institut de recherche de l'École Navale, 29240 Brest Cedex 9, France
{imad.afyouni, cyril.ray, christophe.claramunt}@ecole-navale.fr
[2]University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain
silarri@unizar.es

Résumé : *Les requêtes dépendantes de la localisation peuvent être considérées comme des éléments clés pour le développement des différentes catégories de services géolocalisés et sensibles au contexte. La plupart des travaux sur le traitement des requêtes dépendant de la localisation se concentrent cependant sur les environnements “outdoor”. L'objectif de ce papier est d'étudier les requêtes dépendantes de la localisation et sensibles aux autres éléments du contexte dans les environnements “indoor” (e.g., bâtiments, centres commerciaux, etc.). Cet article présente un modèle de données hiérarchique et sensible au contexte, qui considère différentes dimensions du contexte en plus de la localisation des entités concernées, telles que le temps et les profils des utilisateurs. Un langage de requête continue est également introduit et illustré par des exemples de requêtes, démontrant ainsi le potentiel de la grammaire proposée.*

Mots-clés : Services sensibles au contexte, requêtes dépendantes de la localisation, modèle de données indoor, langage de requête

1 Introduction

Les services géolocalisés ont récemment fait l'objet de recherches approfondies, étant donné que le développement de ces services devrait avoir un impact significatif pour les utilisateurs finaux dans tous les environnements (i.e., indoor et outdoor) [11, 13]. Ces services offrent un accès personnalisé à l'information en tenant compte de la localisation des utilisateurs mobiles. *Les systèmes sensibles au contexte* utilisent différentes dimensions du contexte telles que les dimensions centrées utilisateur (e.g., profil, capacités physiques et cognitives, etc.), le contexte environnemental (e.g., position, entités sociales, lumière, etc.), le contexte temporel et le contexte d'exécution (e.g., topologie du réseau, ressources à proximité, etc.), afin d'anticiper les besoins de l'utilisateur et de personnaliser son expérience [4, 12].

Une intégration réussie des espaces indoor dans les systèmes sensibles au contexte requiert le développement d'un modèle spatial dynamique et flexible qui aidera à fournir des services appropriés aux utilisateurs mobiles se déplaçant dans l'environnement. Dans [1], nous avons proposé un modèle spatial hiérarchique et sensible au contexte qui intègre différents niveaux de granularité et représente : (i) les entités situées ou évoluant dans l'environnement ; une entité peut être une personne (i.e., un utilisateur mobile ou toute autre entité sociale [1]) ou un objet d'intérêt (e.g., capteurs, entrées/sorties, tables, phénomènes continus tels qu'un feu, etc.) ; (ii) leurs propriétés spatiales (e.g., position et emprise spatiale) ; et (iii) les comportements qui s'en dégagent (i.e., comment ces objets peuvent interagir et communiquer au sein de l'environnement).

Les requêtes dépendantes de la localisation (RDL) [11, 17] (en anglais “*location-dependent queries*”) sont des exemples typiques de services “push-based” nécessaires dans les systèmes sensibles au contexte. Dans ce type de requêtes, tout changement de position des objets qui sont impliquées dans la requête peut influencer de manière significative la réponse. Par exemple, si un utilisateur demande à connaître les positions de ses amis dans une zone de 100 mètres tout

1. Les êtres humains situés dans le voisinage de l'utilisateur et qui pourraient influencer la réponse à une requête sont considérés comme des entités sociales.

en se déplaçant dans un centre commercial, la réponse dépendra à la fois de la position actuelle de l'utilisateur ainsi que des positions de ses amis les plus proches. Ce type de requête est particulièrement difficile car, dans la plupart des cas, les positions de l'utilisateur et des entités impliquées dans la requêtes évoluent dans le temps. Une étude exhaustive sur les requêtes RDL a été présentée dans [11] ; certains types de requêtes particulièrement pertinents dans notre contexte sont brièvement décrits comme suit :

1. *Les requêtes de localisation* déterminent les positions des objets d'intérêt, et sont traitées selon un modèle géométrique (e.g., coordonnées cartésiennes ou latitude-longitude-altitude) ou symbolique (e.g., identifiant d'une pièce, d'un étage ou d'un bâtiment) de l'espace (cf., [2]). Ce type de requête est essentiel puisque d'autres requêtes RDL ne peuvent pas être réalisées sans être alimentées par les positions actuelles des objets d'intérêt [3].

2. *Les requêtes de navigation* aident les utilisateurs à trouver et à atteindre des points/objets d'intérêt en leur fournissant des informations de navigation tout en optimisant certains critères tels que la distance totale parcourue ou le temps de déplacement.

3. *Les requêtes de zone* (en anglais "*Range ou Window queries*") sont utilisées pour trouver et récupérer des informations sur des points/objets d'intérêt dans une zone spécifiée par l'utilisateur [16].

4. *Les requêtes de recherche des K voisins les plus proches* ("*K nearest neighbour (KNN) queries*") permettent de sélectionner les K voisins les plus proches de la position actuelle de l'objet mobile [14]. Contrairement aux requêtes de zone, les objets sélectionnés dans les requêtes KNN satisfont une certaine spécification (e.g., l'imprimante couleur la plus proche disponible dans le bâtiment ou les k amis les plus proches).

La gestion efficace des données statiques et dynamiques est un enjeu clé du traitement des requêtes RDL, puisque le résultat d'une requête n'est valable que pour une position particulière de l'objet référence, et pour certaines positions des objets d'intérêt. Comme ces requêtes sont sensibles au temps et dépendantes de la localisation, elles doivent être traitées sous forme de *requêtes continues* [15], ce qui signifie que le système doit maintenir les réponses à jour, jusqu'à ce que la requête soit explicitement annulée par l'utilisateur. Bien que de nombreuses études ont examiné les requêtes RDL, peu de travaux ont abordé l'intégration d'autres dimensions du contexte, particulièrement celles liées à l'*utilisateur* ainsi que le *contexte environnemental*, lors du traitement de la requête. En effet, cela peut complètement modifier la réponse à une requête, même si les positions des objets concernés n'ont pas changé.

Cet article étudie les systèmes sensibles au contexte ainsi que les requêtes RDL dans les environnements indoor, en accordant une attention particulière aux requêtes de navigation, de zone et de recherche des K voisins les plus proches. Une combinaison unique de défis se pose, étant donné que l'approche doit être en mesure de représenter différents types de requêtes de manière flexible, et de prendre en compte des éléments de contexte supplémentaires, ainsi que la structure hiérarchique de l'environnement. La suite de cet article est organisée de la manière suivante. La section 2 présente le modèle de données spatiales, en mettant l'accent sur sa structure hiérarchique, et tout en affirmant son intérêt pour les services géolocalisés. La section 3 propose un langage de requête pour exprimer les requêtes RDL continues dans un environnement indoor, ainsi que quelques exemples de requêtes qui montrent le potentiel de la grammaire proposée. La section 4 conclut l'article et expose les directions dans lesquelles nos futurs travaux peuvent être poursuivis.

2 Approche de Modélisation

Les modèles spatiaux en indoor ont été étudiés et développés dans de nombreux domaines (e.g., robotique mobile, systèmes d'information géographique et l'informatique ambiante). Les modèles de données symboliques orientés graphe fournissent des solutions plus pratiques pour

calculer un itinéraire optimal et réaliste vers une destination en tenant compte des contraintes architecturales et des changements dynamiques dans l'environnement. Des modèles de graphes (réseaux spatiaux) dynamiques sensibles au temps ont été récemment proposés dans [7, 8]. Des modèles hiérarchiques de données spatiales ont également été présentés pour pallier aux problèmes de performance et de passage à l'échelle lors de l'exécution des recherches de chemin sur des graphes de grande taille.

Dans [2], nous avons introduit des besoins préliminaires pour le développement d'un modèle de données spatiales en indoor qui soit le mieux adapté aux systèmes de navigation sensibles au contexte. Ceux-ci sont classées en deux catégories : des besoins orientés services ainsi que des besoins liés à la performance. Cet article propose une approche de modélisation qui étend notre travail préliminaire rapporté dans [1], et qui introduit une représentation hiérarchique de données spatiales dans laquelle des informations de localisation des objets mobiles peuvent être présentées à différents niveaux d'abstraction.

Ce modèle de données hiérarchique et sensible au contexte d'un environnement indoor se distingue par ses trois composantes complémentaires :

- La composante spatiale est constituée d'un ensemble de couches organisées hiérarchiquement et représentant l'espace indoor. Cette composante intègre aussi la gestion d'autres éléments du contexte, et plus particulièrement, le temps et les profils utilisateurs.
- La deuxième composante englobe les entités (i.e., personnes et objets d'intérêt) situées ou évoluant dans l'environnement.
- La composante des actions représente les actions qui sont soit prédéfinies et déclenchées automatiquement par le système d'information sous forme de messages informatifs contextuels, soit générées par des entités données agissant dans l'environnement.

Ces trois composantes sont examinées plus en détail ci-après.

2.1 Composante Spatiale

Dans cette section, les propriétés de différentes couches de la composante spatiale sont décrites, ainsi que leurs utilités pour le traitement des requêtes RDL.

2.1.1 Couche Spatiale de Base

La couche de base du modèle est constituée d'un graphe sensible au contexte à un niveau micro $G_{micro} = (V_{micro}, E_{micro}, W_{length}, W_{time})$ généré à partir d'une grille qui couvre l'espace indoor (Fig. 1). Les noeuds du graphe représentent ainsi les cellules de la grille, et les connexions entre les cellules sont matérialisées par des arêtes.

Dans la définition de G_{micro}, $V_{micro} = \{v_i\}$ désigne l'ensemble de sommets et $E_{micro} \subseteq V_{micro} \times V_{micro}$ est l'ensemble des arêtes. Pour chaque arête $e = (v_i, v_j) \in E_{micro}$, il existe deux fonctions dépendantes du temps $\omega l_{i,j}(t) \in W_{length}$ et $\omega t_{i,j}(t) \in W_{time}$ qui calculent la *distance* et le *temps de parcours* séparant la source v_i de la cible v_j, respectivement, si le parcours du graphe est lancé à l'instant t. En plus du temps, ce modèle prend également en compte d'autres dimensions telles que les *profils utilisateurs*, pour associer davantage des impédances aux arêtes. Les profils utilisateurs sont gérés statiquement afin de dériver des graphes génériques provenant du graphe G_{micro} et qui correspondent à des catégories prédéfinies d'utilisateurs.

Chaque noeud possède un ensemble de propriétés qui maintiennent des informations sur sa position, son statut actuel (i.e., s'il est accessible ou non), un ensemble d'identifiants qui fait référence aux unités spatiales auxquelles le noeud appartient (i.e., identifiants de la pièce, de l'étage et du bâtiment) et l'ensemble d'actions déclenchées en temps réel (i.e., messages contextuels ou notifications) et qui peuvent être exécutées selon certaines contraintes contextuelles (par exemple, pour rappeler un utilisateur qui se déplace dans un centre commercial d'acheter des produits alimentaires ou de fruits quand il se trouve à côté d'un supermarché).

La couche de base est construite d'une première phase hors ligne, et d'une phase ultérieure en ligne qui est en charge de la mise à jour des changements potentiels de données dépendantes

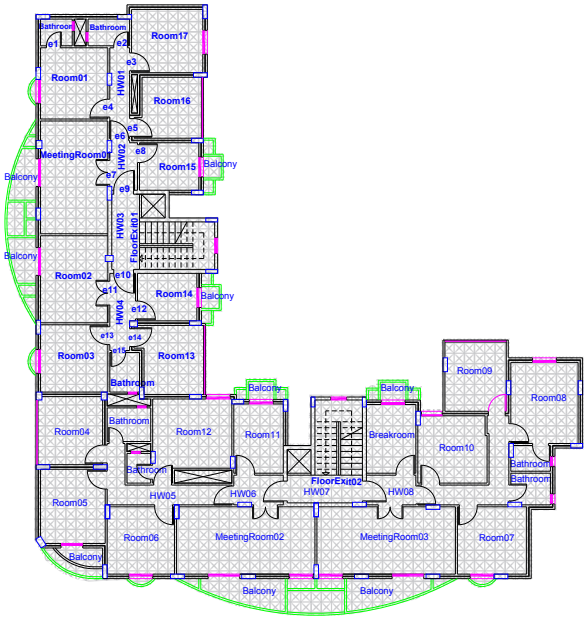

FIGURE 1 – Un graphe à un niveau micro d'un plan d'étage : premier niveau du modèle hiérarchique

du temps. Lors de la première phase, les noeuds qui sont couverts par des objets statiques (e.g., un mur, une table, etc.) sont marqués comme étant occupés alors que le reste est initialement libre. En outre, le statut d'un noeud dépend aussi du profil de l'utilisateur, puisque différents types d'utilisateurs peuvent avoir un ensemble de noeuds accessibles complètement différent (e.g., un certain noeud peut ne pas être occupé physiquement, mais appartient à une pièce qui ne peut être accessible qu'avec une carte-clé). Le seul changement dynamique qui doit être détecté en temps réel fait référence à l'occurrence d'événements qui peuvent avoir un impact direct sur l'accessibilité de noeuds. Un exemple typique est le début d'incendie dans une pièce du bâtiment. Dans cette situation, les avertisseurs d'incendie sont censés détecter cet événement et le communiquer au système. Avec des mises à jour périodiques effectuées automatiquement, le système serait capable de représenter l'emprise spatiale croissante de l'incendie, et de marquer ainsi les noeuds de cet espace physique comme inaccessible aux utilisateurs.

2.1.2 Couches Abstraites

Hiérarchie d'entrée/sortie Une entrée/sortie (en anglais *exit*) est un élément essentiel du modèle de données, à travers laquelle un utilisateur peut quitter ou entrer dans un lieu (e.g., portes ou escaliers). Une sortie est représentée comme un noeud abstrait qui appartient à deux unités spatiales différentes, et est dérivée en agrégeant les noeuds frontaliers de ces deux unités adjacentes (les sorties sont illustrées par des e_i sur Fig. 1). Les distances du réseau optimales et les temps de parcours entre les paires de sorties sont pré-calculés et mises en cache afin de réduire le calcul de recherche de chemin en temps réel. Au second niveau d'abstraction, une hiérarchie d'entrée/sortie est par conséquent construite, et qui est utilisée pour le traitement des requêtes RDL qui nécessitent un calcul précis de la distance.

Les sorties dans cette couche sont organisées de manière hiérarchique pour mieux refléter leur sémantique dans le contexte de la navigation indoor [9]. Comme illustré dans Fig. 2, la structure hiérarchique permet de distinguer entre une sortie d'une pièce, *RoomExit*, et une sortie d'un étage, *FloorExit*, qui est représentée à un niveau d'abstraction supérieur en raison de son importance, de sorte qu'un trajet direct à partir d'une position actuelle vers la sortie de l'étage la plus proche peut être facilement déterminé. Il existe aussi d'autres arcs entre les sorties du même niveau en fonction de leur connectivité (liens horizontaux illustrés par des lignes pointillées dans Fig. 2). Il convient également de noter qu'une généralisation de cette

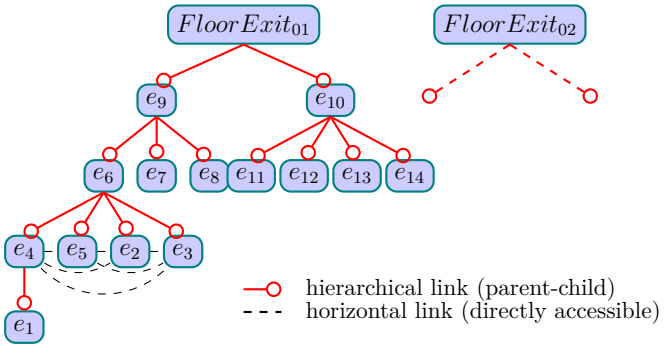

FIGURE 2 – Une partie de la hiérarchie d'entrée/sortie provenant de la couche fine de base (Étage-01, Bâtiment-1)

hiérarchie qui couvre un bâtiment multi-étages est utilisée pour la planification de chemin. Par conséquent, une sortie du rez-de-chaussée fait référence à la sortie du bâtiment, *BuildingExit*, comme un noeud parent, et à la sortie du premier étage comme un noeud fils.

Hiérarchie d'unités structurelles L'intégration des informations sur les sorties dans la hiérarchie topologique permet la modélisation explicite des chemins optimaux à un niveau d'abstraction supérieur. Celles-ci sont utilisées pour faciliter des parcours hiérarchiques du graphe et pour pallier les problèmes de performances lors de la traversée du graphe de base. Cependant, la sémantique topologique telle que la connectivité entre les unités structurelles n'est pas explicitement matérialisée dans la hiérarchie d'entrée/sortie, même si des informations qui représentent leur appartenance à la hiérarchie topologique ont été incorporées. Par conséquent, une hiérarchie d'unités structurelles qui repose sur un graphe de connectivité, représentant les pièces comme des noeuds et les connexions comme des arcs, peut être dérivée comme une couche supplémentaire afin de préserver les relations topologiques (Fig. 3).

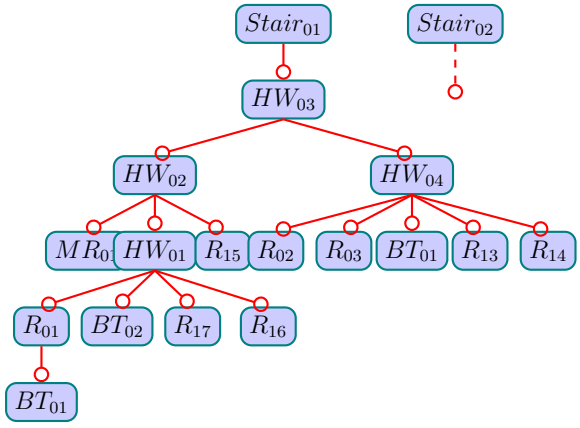

FIGURE 3 – Une partie de la hiérarchie d'unités structurelles provenant de la couche fine de base

Une telle hiérarchie peut être directement dérivée à partir de la couche de base. En effet, le processus d'agrégation, *"clustering"*, consiste en : (1) l'extraction et l'agrégation des noeuds dont l'identifiant de la pièce est identique pour former les nouveaux noeuds abstraits de la hiérarchie ; (2) la détermination des chemins optimaux entre les unités structurelles connectées en utilisant les fonctions dépendantes du temps préalablement définies et (3) la création des arcs abstraits entre les unités connectées en initialisant les poids aux valeurs calculées dans l'étape précédente.
Par conséquent, cette conception hiérarchique est très utile, car elle prend en charge une large gamme d'applications, et permet de pallier les problèmes de performance et de passage à l'échelle lors du traitement des requêtes RDL.

2.2 Composante des Entités

Cette composante représente les personnes et les objets d'intérêt de l'espace indoor. Ces entités sont soit fixées à l'infrastructure (i.e., objets statiques tels que des portes, des murs, des capteurs fixes, etc.) soit dynamique (e.g., utilisateurs mobiles, phénomènes continus). Une entité est caractérisée par des propriétés statiques (i.e., des attributs tels que son type, son statut et une description quantitative et qualitative) et potentiellement d'autres propriétés dynamiques telles que des *espaces d'interaction* (introduits dans [5] et adaptés dans un travail précédent [1]) qui couvrent des informations sémantiques utilisées à des fins d'interaction. Les espaces d'interaction identifiés sont : (1) *l'espace physique* représenté par l'ensemble de noeuds couverts par l'entité à un instant donné ; (2) *l'espace fonctionnel* désigne les noeuds sur lesquels une autre entité peut interagir physiquement avec l'entité considérée ; (3) *l'espace de zone* est un paramètre spécifique uniquement affecté aux capteurs et désigne l'ensemble des noeuds couverts par le capteur ; et (4) *l'espace opérationnel* d'un utilisateur mobile est l'ensemble des noeuds qui lui sont accessibles à un instant donné. De plus, un objet peut effectuer une liste sélectionnée d'actions qui peuvent être déclenchées en fonction de certaines contraintes contextuelles dépendantes de l'application.

2.3 Composante des Actions

Cette composante modélise l'ensemble des actions qu'une entité donnée peut effectuer. Les actions sont dépendantes du contexte ; ce qui signifie que, à un instant donné et pour une certaine entité, seulement une liste spécifique d'actions possibles est valide. Pour un utilisateur mobile, les actions comprennent une séquence de mouvements, des interactions avec d'autres entités voisines et avec des objets situés dans l'environnement, un certain nombre d'interrogations sur des services en vue d'atteindre un objectif prédéfini. L'approche présentée permet de modéliser efficacement des objets d'intérêt situés dans l'environnement, afin que les utilisateurs qui sont engagés dans une activité donnée puissent recueillir des connaissances et mieux comprendre leur entourage. Un utilisateur peut ainsi reconfigurer et manipuler des objets physiques (e.g., une chaise, une porte, un chauffage, etc.) ou d'autres objets virtuels (e.g., une image 2D/3D d'un objet physique, une interface utilisateur numérique, des recommandations/-informations, etc.) afin de produire des changements dans l'environnement. Un utilisateur peut de plus communiquer avec tout capteur fixe ou mobile situé dans l'espace de zone du capteur mobile qui lui est rattaché ou qui est intégré dans son appareil (e.g., un capteur MEMS, une étiquette RFID, etc.). Lorsque l'on considère les phénomènes continus, leurs actions peuvent se matérialiser par la façon dont un phénomène donné se diffuse dans l'espace.

3 Requêtes RDL dans les Environnements Indoor

Comme indiqué précédemment, certains types de requêtes RDL présentent un intérêt particulier dans les environnements indoor, telles que les requêtes de navigation, de zone et de recherche des K voisins les plus proches. Afin d'améliorer l'expressivité des requêtes, une grammaire est introduite dans cette section. Cette grammaire est une extension d'un travail précédent [10] adaptée au contexte des environnements indoor, et qui supporte les requêtes de navigation tout en favorisant d'autres préférences spécifiques au modèle de requête (Fig. 4). Par exemple, elle inclut des opérateurs (e.g., *All-routes*) et des contraintes (e.g., *Stop-vertices*) utilisées dans les requêtes de navigation et inspirées par [6].

Dans la structure générale du langage de requête, deux types de requêtes sont identifiés : le premier représente typiquement une structure standard de requête SQL (*Standard-query*) ainsi que des types spécifiques de contraintes dépendantes de la localisation (*inside* et *nearest*), qui sont essentiellement utilisés pour exprimer les requêtes de zone et de recherche des voisins les plus proches. Le second représente les requêtes de navigation qui ont besoin d'intégrer un

General query structure

Query	→	(Standard-query | Navigation-query)
Standard-query	→	*select* (Attr-Projections | '*') *from* Class-names (*where* Conds) ?
Navigation-query	→	*select* (Attr-Projections | '*') *from* All-routes-expression (',' Class-names)* (*with* Stop-vertices) ? (*where* Conds) ? (*optimization-criteria*) ?
Attr-Projections	→	Attr-Loc-Select (',' Attr-Loc-Select)*
Attr-Loc-Select	→	attribute | Loc-Select
Loc-Select	→	Object-id ':' 'loc' | *gr* '('Map-id ',' Class-name ')' | *gr* '('Map-id ',' Route-id ')'
All-routes-expression	→	*All-routes* '(' Loc-Ref ',' Loc-Target ')'
Loc-Ref	→	Object-id (':' 'coord') ? | *gr* '('Map-id ',' Object-id ')' | *gr-map* '('Map-id ',' Gr-id ')' | Vertex-id
Loc-Target	→	Class-name | Object-id | Vertex-id ':' 'coord' | *gr* '(' Map-id ',' Class-name ')'
Stop-vertices	→	Stop-vertex (',' Stop-vertex)*
optimization-criteria	→	(*minimize* | *maximize*) Measure
Measure	→	*time* | *distance*

Conditions can be standard conditions on attributes or location-dependent conditions

Cond	→	(Bool-Cond | LDQ-Cond)
Bool-Cond	→	attribute Comp Value | *intersect* '(' Vertex-set ',' Vertex-set ')' | Value IN Vertex-id ':' POI
LDQ-Cond	→	*inside* '(' Args-Inside ')' | *nearest* '(' Args-Nearest ')' | ...
Args-Inside	→	Radius ',' Loc-Ref ',' Loc-Target
Args-Nearest	→	K ',' Loc-Ref ',' Loc-Target
Radius	→	Real Units

FIGURE 4 – Une partie de la grammaire utilisée pour les requêtes RDL dans les environnement indoor

mécanisme de calcul d'itinéraire lors du traitement, tout en optimisant les critères de la distance et/ou du temps. Pour une requête de navigation, la ⟨ FROM clause ⟩ contient un appel externe à l'opérateur *All-routes*, qui a la syntaxe générale suivante : *All-routes (Loc-Ref, Loc-Target)*. Cet opérateur retourne un ensemble de routes valides entre la position actuelle de l'objet référence et celle de l'objet cible. Les arguments *Loc-Ref* et *Loc-Target* peuvent correspondre soit à un identifiant d'un noeud "Vertex-id" soit aux positions actuelles de l'objet référence et de l'objet cible, respectivement. *Loc-Target* peut aussi faire référence au nom de la classe de l'objet cible ; ceci est utilisé, par exemple, dans la contrainte *inside* afin de récupérer tous les objets d'un type donné.

La ⟨ WITH Stop-vertices clause ⟩ indique que la route doit passer par un endroit qui a un intérêt pour l'utilisateur. Par ailleurs, deux critères d'optimisation sont appliqués : *time* et *distance*, qui sont considérés en se basant sur les fonctions définies précédemment ($\omega l_{i,j}(t)$ et $\omega t_{i,j}(t)$). Dans la structure standard, deux types de conditions dépendantes de la localisation peuvent être exprimés dans la ⟨ WHERE clause ⟩ : *inside(Radius, Loc-Ref, Loc-Target)* et *nearest(K, Loc-Ref, Loc-Target)*. Une contrainte *inside* est appliquée lors de l'exécution d'une requête de zone, et prend en compte le seuil maximal à examiner, pour calculer l'ensemble de chemins (autour de l'objet référence). La contrainte *nearest* est exprimée pour exécuter les requêtes de K voisins les plus proches, en précisant le nom de la classe d'objets d'intérêt dans Loc-Target.

Le concept du *location granule* proposé dans [10] est adopté. Un granule regroupe un en-

semble de positions à un niveau plus fin (dans notre cas, les coordonnées des noeuds dans le graphe de base) sous un nom commun. L'utilisation des granules permet de formuler des requêtes avec une résolution spatiale appropriée à l'application prévue (e.g., des noeuds à un niveau micro, pièces, étages, bâtiments, etc.). L'opérateur *gr* est un raccourci de *granule* et retourne le granule associé à l'objet selon le niveau de granularité choisi. Ce concept de granules est étendu pour être utilisé au niveau de la clause SELECT afin d'illustrer les routes de navigation à un niveau d'abstraction choisi, ainsi qu'au niveau de la clause FROM pour gérer les requêtes de navigation.

Comme le montre la figure 4, l'opérateur *gr* peut être référencé dans la clause SELECT, la clause FROM et/ou la clause WHERE de la requête, en fonction de l'utilisation des granules pour la visualisation des résultats et/ou le traitement des contraintes ou le calcul des itinéraires. Pour la visualisation, cet opérateur est utilisé comme une projection *Loc-Select* dans la clause SELECT, selon la demande soumise par l'utilisateur, pour afficher le résultat au niveau de granularité désiré ; par exemple, SELECT gr('room-level', Person) est utilisé pour projeter les pièces où les personnes récupérées par la requête sont situés. L'opérateur *gr* peut en plus être appliqué sur une route, qui est le résultat d'une requête de navigation, afin de montrer la séquence de noeuds et d'arêtes obtenues dans la route à un niveau d'abstraction choisi. Ainsi, SELECT gr('room-level', Routes.id) sera utilisé pour illustrer la séquence de pièces inférée de l'itinéraire composé initialement de noeuds et des arêtes de différents niveaux de granularité (e.g., niveau micro et la hiérarchie de sortie).

En revanche, le même opérateur *gr* peut être spécifié pour le traitement des requêtes comme un argument *Loc-Ref* et/ou *Loc-Target* de la clause FROM (i.e., dans l'expression *All-routes-expression*), et/ou dans les contraintes (i.e., inside et nearest), en référence aux positions des objets concernés. Par exemple, *inside(100 meters, gr('room-level', 'o1'), Person)* est une contrainte satisfaite par les personnes qui se situent sur une route valide (i.e., distance inférieure à 100 mètres) autour de la pièce où l'objet o1 est situé. Au contraire, *inside(100 meters, 'o1', Person)* serait utilisé lorsque la zone souhaitée est déterminée autour de l'objet *o*1 lui même.

3.1 Exemples de Requêtes Dépendantes de la Localisation

Cette section présente quelques exemples de requêtes RDL qui montrent le potentiel de la grammaire proposée. En particulier, nous considérons des requêtes de navigation, des requêtes de zone et des requêtes de recherche des voisins les plus proches.

3.1.1 Requêtes de Navigation

Les requêtes de navigation aident à découvrir des chemins optimaux vers un point/objet d'intérêt, tout en respectant les contraintes dynamiques de l'environnement. Quelques exemples de requêtes de navigation sont présentés ci-dessous.

1. Un utilisateur identifié par 'userID' demande de trouver le chemin le plus rapide de sa position actuelle vers la salle de réunion 'MR01' qui passe à travers une salle de pause 'break-room' ; Afficher le résultat à un niveau *room level* :

```
SELECT gr('room-level', RO.id)
FROM Room AS R, Person AS P,
All-routes(gr('micro-level',P),R)AS RO
WITH Stop-vertices v1
WHERE R.id = 'MR01' AND P.id ='userID'
AND 'break-room' IN v1.POI
MINIMIZE time(RO)
```

où time(R) = $time_{start \to goal}(t_{start})$ est le temps estimé pour traverser le chemin R de 'userID' situé au noeud v_{start} vers 'MR01'.

2. Trouver le plus court chemin de la personne 'userID1' vers la personne 'userID2'; Afficher le résultat à un niveau *room level* :

```
SELECT gr('room-level', RO.id)
FROM Person AS P1, Person AS P2
All-routes(gr('micro-level', P1),
    gr('micro-level', P2)) AS RO
WHERE P1.id = 'userID1'
AND P2.id = 'userID2'
MINIMIZE length(RO)
```

où length(R) = $length_{start \to goal}(t_{start})$ est la distance de la route dépendante du temps de 'p1' situé au noeud v_{start} vers 'p2' situé au noeud v_{goal}.

3.1.2 Requêtes de Zone

Les requêtes de zone sont utilisées pour récupérer des informations sur les objets ou les lieux qui se trouvent dans la zone spécifiée. Des exemples de telles requêtes sont :

1. Récupérer les identifiants des personnes dans une zone de 100 mètres autour la pièce où l'objet $o1$ est situé :

```
SELECT Person.id
FROM Person
WHERE inside(100 meters,
      gr('room-level', 'o1'), Person)
```

2. Récupérer toutes les entités communicantes dans le voisinage (à une distance inférieure à 100 mètres) d'un utilisateur identifié par 'userID' et avec une portée de communication d'au moins 100 mètres :

```
SELECT CO.id
FROM Object AS CO
WHERE inside(100 meters,
      gr('micro-level','userID'), CO)
AND CO.Communicate = true
AND CO.commRange >= 100
```

3.1.3 Requêtes de K Voisins les Plus Proches

Une requêtes de (K) voisins les plus proches récupère les (K) objets qui satisfont certaines spécifications et qui sont les plus proches à un objet ou un endroit.

1. Trouver la plus proche salle de bains disponible pour l'utilisateur identifié par 'userID' :

```
SELECT BR.id
FROM Bathroom AS BR
WHERE nearest(1, gr('micro-level',
      'userID'), BR)
AND BR.state = 'free'
```

2. Trouver les deux imprimantes couleur les plus proches à chaque membre du département informatique :

```
SELECT Pr.id, P.id
FROM Printer AS Pr, Person AS P
WHERE
nearest(2, gr('micro-level', P), Pr)
AND 'C.S. Department member' IN P.FD
AND Pr.type = 'ColourPrinter'
```

où FD "*Feature Description*" correspond aux propriétés statiques d'écrivant l'objet d'intérêt.

En résumé, le langage de requête favorise les requêtes RDL orientées navigation et intègre d'autres préférences dans le modèle de requête. De plus, ce langage gère la granularité des positions des objets statiques et mobiles, favorisant ainsi le modèle hiérarchique de données présenté précédemment.

4 Conclusions & Perspectives

Cet article présente une approche pour modéliser les requêtes dépendantes de la localisation dans les environnements indoor. Un modèle de données hiérarchique et sensible au contexte est présenté, qui mène à l'utilisation d'autres dimensions du contexte en plus de la localisation des entités concernées, telles que les profils utilisateurs et le temps. La structure hiérarchique de ce modèle favorise un traitement adaptatif et efficace des requêtes RDL. D'autre part, un langage de requêtes qui permet de représenter une variété de requêtes RDL est introduit. La grammaire proposée intègre les préférences des utilisateurs (e.g., points d'intérêts intermédiaires, choix d'optimiser le calcul du chemin selon le temps ou la distance) et d'autres sémantiques (e.g., génération des graphes adaptés aux profils utilisateurs). En outre, l'utilisation des granules dans le langage augmente de façon significative l'expressivité des requêtes.

Les futurs travaux seront orientées vers : (1) l'implémentation des opérateurs définis dans la Section 3 (i.e., l'opérateur *All-routes*, et les contraintes *inside* et *nearest*). En effet, deux algorithmes prometteurs pour le traitement continu des requêtes de navigation et de zone ont déjà été développés ; (2) l'intégration d'un modèle de contexte étendu avec un raisonnement sémantique en temps réel, principalement pour la gestion des événements en temps réel ; et (3) la généralisation du modèle de données hiérarchique aux niveaux d'abstraction supérieurs (niveau d'étage et de bâtiment).

5 Remerciements

Les travaux présentés dans cet article ont été réalisés dans le cadre d'une mission scientifique (STSM) financée par l'Action MOVE COST IC0903 et du projet CICYT TIN2010-21387-C02-02. Les auteurs tiennent à remercier ces soutiens financiers.

Références

[1] I. Afyouni, C. Ray, and C. Claramunt. A fine-grained context-dependent model for indoor spaces. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 33–38. ACM, 2010.

[2] I. Afyouni, C. Ray, and C. Claramunt. Spatial models for indoor and context-aware navigation systems : A survey. *Journal of Spatial Information Science (JOSIS), accepted*, pages 1–43, 2012.

[3] C. Becker and F. Durr. On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9(1) :20–31, 2005.

[4] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2) :161–180, 2009.

[5] M. Bhatt, F. Dylla, and J. Hois. Spatio-terminological inference for the design of ambient environments. In *Proceedings of the 9th International Conference on Spatial Information Theory (COSIT)*, pages 371–391. Springer, 2009.

[6] J. Booth, P. Sistla, O. Wolfson, and I. Cruz. A data model for trip planning in multimodal transportation systems. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT)*, pages 994–1005. ACM, 2009.

[7] D. Delling. Time-dependent SHARC-routing. *Algorithmica*, 60(1) :60–94, 2011.

[8] B. Ding, J. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*, pages 205–216. ACM, 2008.

[9] H. Hu and D. Lee. Semantic location modeling for location navigation in mobile environment. In *Proceeding of the IEEE International Conference on Mobile Data Management (MDM)*, pages 52–61. IEEE, 2004.

[10] S. Ilarri, C. Bobed, and E. Mena. An approach to process continuous location-dependent queries on moving objects with support for location granules. *Journal of Systems and Software*, 84(8) :1327–1350, 2011.

[11] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing : Where we are and where we are heading. *ACM Computing Surveys*, 42(3) :1–73, 2010.

[12] M. Petit. *Approche spatiale pour la caractérisation du contexte d'exécution d'un système d'information ubiquitaire*. PhD thesis, Arts et Métiers ParisTech - Institut de Recherche de l'Ecole Navale, 2010.

[13] J. Schiller and A. Voisard. *Location-Based Services*. Morgan Kaufmann, San Francisco, CA, USA, 2004.

[14] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 287–298. VLDB Endowment, 2002.

[15] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 321–330. ACM, 1992.

[16] K. Wu, S. Chen, and P. Yu. Incremental processing of continual range queries over moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 18(11) :1560–1575, 2006.

[17] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. Lee. Location-based spatial queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 443–454. ACM, 2003.