Middleware-free Approach for Indoor Space Shortest Path Queries

Master thesis
de101f13

Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg

Aalborg University

Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg
Telephone 99 40 99 40
http://www.cs.aau.dk

Title:
Middleware-free Approach for Indoor Space Shortest Path Queries

Theme:
Indoor Navigation

Semester:
Data Engineering
10^{th} semester
February. 3^{rd}, 2013 - Sep. 13^{th}, 2013

Group:
de101f13

Members:
Aistė Pilvinytė

Supervisor:
Hua Lu

Copies: 2

Report - pages: 35

CD-ROM: 1

Total pages: 35

Abstract:

This report is the documentation of the master thesis project that pursues to show that indoor space shortest path queries processed directly on the database engine without any middle-ware are effective.
The project includes the usage of indoor space data model and it's adaptation to support effective shortest path queries in indoor spaces. The model takes partitions, doors and connectors with their geometrical values as objects and provides the connections between those elements with the help of spatial properties only. In order model to increase the efficiency of the queries the model was extended by R-Tree indexing structure.
Finally, the number of queries were written to show, that the model is able to support the queries without the help of aditional middle-ware tools and it is efficient.

Preface

This report was written during the 4th Data Engineering semester at the Department of Computer Science, Aalborg University. The project started at the 2nd of February 2013, and ended at the 13th of September 2013.

Conventions used in the report are as follows:

- Citations are indicated by square brackets surrounding one or more digits either at the end of the sentence, before the period, or drectly after the part that is cited.

- Abbreviations appear in their extended form in the first use and short in the following appearances. Some abbreviations are deemed so conventional that presenting them in their extended form is unnecessary.

- In order to improve the readability of the report, pronouns are used irrespectively of gender. For instance, "he" refers to he/she.

- "We" and "our" refer to the authors of the report.

- AAU refers to the Department of Computer Science at Aalborg University.

Appendices are attached at the end of the report and the electronic version of the report willbe available for other studentson the AAU electronic library: http://projekter.aau.dk/projekter/

The additional files are uploaded here: The md5 hash tag is: 67BA78C984513310170DEF5BB23EF286

Acknowledgements are given to the following people:

- Hua Lu for supervising the group during the project.

Signatures

Aistė Pilvinytė

Date: September 13. - 2013

Contents

List of Figures

Acronyms

LBS Location Based Service

IFC Industry Foundation Class

BIM Building Information Modeling

MBR Minimal Bounding Rectangle

ER Entity-Relationship

Chapter 1

Introduction

The recent years have witnessed an increasing interest in Location Based Services (LBSs) - the services that exploit the knowledge about the location of a service user and uses it in order to create the additional value to them. Such services, that had been first introduced in military 30 years ago, are now widely spread around the world. The development of mobile networking technologies, the reduction of mobile devices scale, and increased number of users made it available for almost everyone.

The idea behind LBS is the integration of information collected from mobile device, e.g. location, and other data from sources [8]. The other data could include user preference (e.g. user is vegetarian), database-specific context (e.g. meal price, waiting time, etc), environmental (e.g. additional road path information) or even combination of all three. The example of such services could be an application that guides a customer in the supermarket according to their general preferences, e.g. customer is a vegetarian, looking for a section of non-meat products, that is closest to their current position.

The majority of LBS are developed for outdoor spaces, even though people tend to spend bigger part of their lifes indoors. This happened for a reason, that positioning technology, which is the essential part of both indoor and outdoor LBSs, was developed only for outdoor spaces, but not suitable for indoors (GPS technology do not work indoors). And this reason was enough to stop development in indoors. But since the situation changed ([10]), the solution area of application is still sparse, therefore, there is a great need to create new services and improve the existing ones for the indoor space.

The large and complex indoor spaces, e.g. airports, schools, shopping malls, can be very challenging to navigate within. This problem can be solved with a help of LBS. The common main task for such services is to find the shortest path from the location of a user to the wanted destination. Navigation queries are the ones, that can give the answers to the questions like "How to reach certain point of interest". The only problem is to figure it out, how to make a query, so the result is the shortest path, and the time spent for processing this query is the shortest. The topology of indoor spatial areas is complex, and there has to be a simple and precise model to define the relationships between the spatial objects. Then the way to measure the distance has to be decided. The outdoor spaces usually defined in Euclidean space (geometric space), where the proximity is measured by Euclidean distance. However, it is not always possible within indoor spaces, and network-based shortest path has to be taken in consideration. The proximity itself can be measured as spatial or temporal distances in network-based topology.

As it was mentioned, there are many different ways to measure the proximity in indoor spaces and evaluate the shortest path. In this project, the shortest path is considered as the shortest in the terms of smallest amount of jumps to be taken in order to find the shortest path. The methods that were used for outdoors are not suitable for indoors, since the natures of environments are too different with the constraints and different data models, structures, proximity evaluations are needed. Therefore, we are proposing a new database oriented solution that is middleware-free to find the shortest paths in indoor spaces. The solution is considered to help avoiding overhead in continuos queries, that happens because of the fact, that position change requires recalculation of the path. The solutions is using R-tree indexing mechanism for optimizing the queries and reaching higher performance.

The paper is organized as follows: chapter 2 describes the related works that his project is based on. Then the chapter 3 gives the description of the project and chapter 4 will describe the approach, that includes a hybrid model and indexing structure proposed by this paper, and, finally, 5 gives some queries and results of the project. The conclusion in 6 gives a short overview.

Chapter 2

Related work

This chapter will give an overview of the research domain area that is related to the project. Thie section 2.1 covers the significant problems and solutions of indoor space modeling, indexing and querying. The following section 2.2 will introduce reader into work, that had been done in the previous semester, since this project is the extension of it.

2.1 Related research

This section gives a brief summary of researches that had been done in the domain area of indoor space data modeling, indexing and querying.

2.1.1 Data modeling

The researches on the topic of indoor spaces are mostly focused on space modeling so far, ever since the notion of indoor space was introduced as an extension for outdoor space [7]. The application of the models that were used for outdoor space was not possible, because indoor and outdoor spaces have diffirences that are critical for modeling.

The differences between indoor and outdoor spaces lie in the perspective of the constraints. The Euclidean space that is widely used for outdoor spaces has none of them. The objects in the space are represented as shapes in coordinates system. On the contrary, the nature of indoor space is highly limited by architectural objects, e.g. doors, walls, corridors. The constraints also make it impossible to measure direct spatial distance, and it has to be replaced with network-based spatial, temporal distance or some other kind

of distance.

The models of space are usually classified as geometrical and symbolic. But indoor space is usually related with the notion of the cellular (symbolic) space, since it's more convenient to ask queries in a manner of "How many people are in room XXX", than to make a query with a coordinates (x_1, y_1) and (x_2, y_2) [7]. In addition, there is also a group of hybrid models, where symbolic and geometric models complement each other.

Despite the fact, that indoor and outdoor spaces are significantly different from perspective of constraints, [6] presented a model that tried to generalize both spaces. This semantic model captures the topology and dynamics of the spaces by representing the space by locations, connecting points, moving objects and routes, and uses them in order to construct a graph. The model does not include any kind of distances into considerations, but still can be used to find a database related shortest path.

Most of the semantic models are graph-based like the works of [6, 11, 1]. Also the topological-based structures and hierarchies are used to capture the connectivity and reachability between spatial units. The reason why models are more convenient to use is that object location is provided semantically using human-readable description is user friendly for spatial querying [2].

The approach of [11] uses link/node based model. The locations as nodes are placed in 2D environment, but there are also vertical conections - edges - between floors, which makes the model 3D based. In order to find the shortest path, that is defined as function, that evaluates the shortness of path takes parameters of length, speed, access and the type-of-person, Dijkstra algorithm is employed.

Feature-based models have a number of problems [1]. At first, they are ineficient in maintaining consistency of topological relationships and finding routes, as connectivity is not presented clearly. Also, for complex structure a big storage is needed. Network-based topology data based on graph theory had been raised to make an improvement. Even complicated real life structures are expressed in quite simple models.

2.1.2 Indexing and querying

The efficient processing of spatial queries relies on indexing structures [2]. Since indoor spaces contains both topological and geometrical data there is a need to index both of them. Inefficient structure causes overhead in

finding spatial objects, which is not easy from a perspective that it is based on the network-based proximity.

The requirements for spatial indexer were presented in [5]. According to the paper there are four requirements for event-based location-aware system in indoor spaces:

- The indexer must generate appropriate room or region change events based on point location data;
- The indexer must generate events describing ingress, egress, and overlap of interaction zones;
- The indexer must support "Nearest neighbour" queries for point data;
- The indexer must account for immutable physical boundaries (walls, etc.) when generating spatial events;

The other researchers made a study of R-trees in [9] and concluded that even though R-tree can be operated only on rectangular regions, it is fast and efficient for general spatial searching. And the combination of R-tree and Quad-tree as in [5] fits given requirements well.

[4] studied the relationships between indexed Minimal Bounding Rectangle (MBR) for using of topological relationships such as overlap, inside, contains, etc. The efficiency of using different databases with more than 10,000 objects, with different sizes of MBR, and 100 queries had been tested and concluded that R-trees are highly efficient for indexing spatial data.

2.2 Previous project

As it was stated before, this project is the second part of the project that was divided between two semesters. This section will focus on the problem found and results achieved during the first semester.

The main goal of a project was to build a prototype for indoor spaces data extraction and interpretation. The data was received in Industry Foundation Class (IFC) format, which is commonly used format for Building Information Modeling (BIM). Initially, we were working on building a data model for indoor spaces. The model had to support routing, accessibility and accessibility rules in terms of indoor spaces elements, such as rooms,

doors and elevators/stairs. These elements were transformed into partitions, access points and connectors respectively. The model was also built to support distances. However, since paths in indoor spaces are not explicit, the model contains information only about distances between access points, which means, that specific location in the partition is not encountered in calculation. The database schema that supports our data model is given in 2.1.

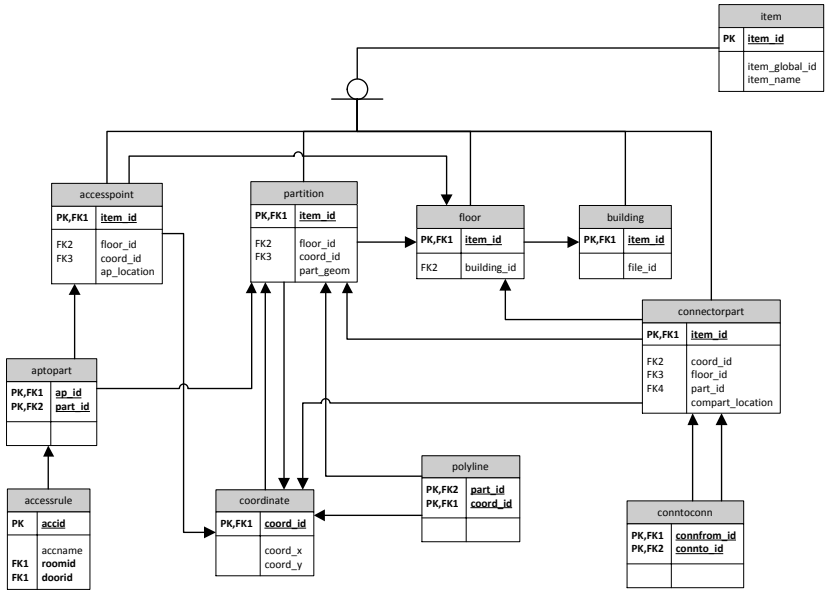

Figure 2.1: *This is an ER diagram illustrating the database design.*

As mentioned, the data used for a project was received in a form of IFC files. The files contains information about rooms, doors, elevators, stairs, etc., but there were no high level relationships between objects. Therefore we extracted and interpreted the data to fit our data model.

The results of the project is the prototype for data extraction and valuable data in understandable format. The quality of data was evaluated by measuring hit-rate. The prototype worked 100% correct while extracting partitions and doors. However, there were some mapping missing, due to errors in some IFC files. But all in all, it is a great database of data for future improvements and extension.

Chapter 3

Project description

This chapter will give a brief description of the project: the explanation of main idea, methods used to achieve it and some information about data.

3.1 Idea elaboration

As it was stated in introduction, even though the computers and mobile devices are getting smaller, but the requirements for software running on those devices are still high and getting even higher. What was acceptable two years ago, now is not. Especially, when we talk about performance and speed requirements.

The idea of the project is to enable the direct shortest path queries on spatial database, that contains data of indoor spaces. By term 'direct' in the definition we mean, that those queries are processed in database processing unit without interference of any additional middleware framework for storing or processing the data (e.g. Java programming language tools like arrays or etc.).

As it was stated in section 2, the indoor and outdoor spaces are very different from perspective of complexity of topology and the same solutions are not applicable for indoor spaces. Therefore, behind this idea, there is a spatial indoor space model, based on the project of previous semester [3], which is a bit too complex for quick queries and for this reason it was extended to better support navigation queries. The database with the spatial extension will enable spatial functionalities such as intersection, adjacency in order to make the processing of spatial objects easier and the inner database

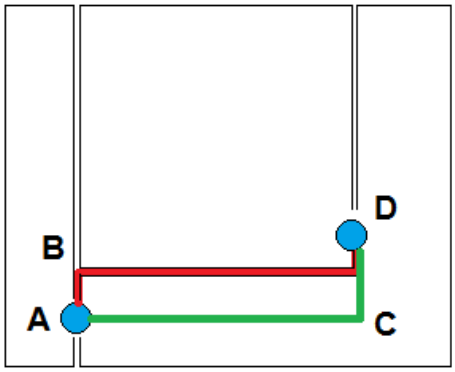

Figure 3.1: *Calculation of distance by Manhattan distance*

indexing structure based on R-tree complements it, by making the queries faster and more efficient.

The shortest path in this project is database related and is defined like this:

Definition 3.1.1. The shortest path is the path between two partitions such that the number of jumps in the database is the least possible. The jump refers to a number of partitions in transit on a path. Basically, it is the depth of the search.

It also includes counting of real distance, based on Manhattan distance. Manhattan distance is chosen because big indoor spaces are rarely staying empty, so it is possible to take the straight path accross the room. And if there is a number of objects to avoid, it will be closer to Manhattan's distance and easier to calculate.

In the example on figure 3.1 there are three rooms: the room number 1 - with the door marked by blue circle A, the room number 2 - with the door marked by blue circle D and the room number 3 - with two doors A and D, that connects the room with previously mentioned locations. In order to calculate the distance between the doors A to door D, several pathes can be taken: through points A, B, D or A, C, D. But the distance would be the same.

3.2 Problem statement

This section describes the problem this project attempts to solve.

To enable direct search of the shortest paths (minimal number of joins) between chosen regions using middleware-free approach (directly to database). This is achieved by:

- Define a model that is using spatial data.
- Implement the model on spatial database.
- Extend database model with R-tree indexing framework.

3.3 Data sources

```
762421= IFCLOCALPLACEMENT(#618550,#762418);
762424= IFCSPACE('2dK70Ro9iaIxjFV7ygtyFe',#13,'0.2.37',$,$,#762421,#762406,'Toilet',.ELEMENT.,.INTERNAL.,$);
762436= IFCCARTESIANPOINT((0.,0.));
762440= IFCCARTESIANPOINT((1470.,0.));
762444= IFCCARTESIANPOINT((1470.,2004.0529));
762448= IFCCARTESIANPOINT((0.,2004.0529));
762452= IFCPOLYLINE((#762436,#762440,#762444,#762448,#762436));
762456= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#762452);
762457= IFCAXIS2PLACEMENT3D(#40,#36,#28);
762460= IFCEXTRUDEDAREASOLID(#762456,#762457,#36,2700.);
762463= IFCSHAPEREPRESENTATION(#882,'Body','SweptSolid',(#762460));
762469= IFCPRODUCTDEFINITIONSHAPE($,$,(#762463));
762473= IFCDIRECTION((-1.,0.,0.));
762477= IFCCARTESIANPOINT((31885.,40240.,0.));
762481= IFCAXIS2PLACEMENT3D(#762477,#36,#762473);
762484= IFCLOCALPLACEMENT(#618550,#762481);
762487= IFCSPACE('0FkHnQ1QyUGwrcv1vtPvWf',#13,'0.2.38',$,$,#762484,#762469,'Forrum',.ELEMENT.,.INTERNAL.,$);
```

Figure 3.2: *The example of IFC file*

The data that will be used for this project is extracted from a number of IFC files using a prototype software that was created in a previous semester. As it was mentioned in our project [[3]], there were a number of mistakes in IFC (example can be seen in figure 3.2) files and therefore in the data extracted from the files the same mistakes still existed, while using those data some information was edited manually, in order to generate right results.

The data source contains data about large indoor spaces witch varies in terms of size, number of floors and complexity . Complexity can be measured by number of paths that can be taken to access one region from other. The relation of partitions can be expressed as non-deterministic graph, that might contain cycles, therefore it is important to make sure, that the algorithms do not get into forever loops.

Chapter 4

Project

This chapter gives the description of what this project is about is about: the data model, the r-tree indexing schema and the queries, that answers to interesting questions.

4.1 Data model

The data model created in last semester (figure 2.1) is mostly for storing data from IFC and keeping the relationships between partitions using access points and connectors while keeping geometrical inforation. The model works well for keeping it and providing the data for applications that are used to draw the mapping of those partitions and access points. But for further analysis and usage of this information a more simple and easier accessible data model is needed. Therefore we are proposing a bit simplier model.

The essential objects of this data model is obviously partition, access point and connector. But there are also other strucutures that helps to keep the relationships more clear: building and floor.

- The object of *partition* - represents the object of the room - the part of building or floor that is limited from other rooms by walls. In the database it is represented by a geometrical figure polygon consisting of a number of connected points.
- The object of *accesspoint* - represents the object of door or any other object that connects precisly two partitions (could be one, if the objects represent the exit doors in the building).

- The object of *connector* represents the object of stairs or elevator. Basically, it is access point, that connects partitions that are assigned to different floors of the same building. Generally,

- The object of $floor$ represents the collection of partitions. The doors, however is not included into the floor, as it it is not important to keep track of them.

- The object of *building* represents the collection of the floors, and then consequently - partitions.

The connections between those objects can be expressed with a diagram (figure 4.1):

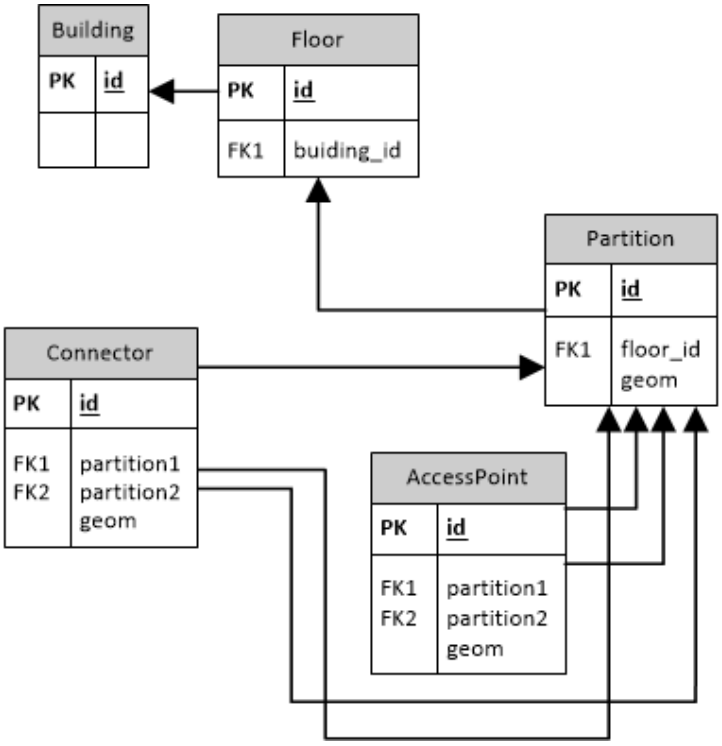

Figure 4.1: *The entity relation schema of data model*

It is important to notice that objects of the *connector* and the $floor$ are only partially connected. The *connector* only connects two objects of *partition* and it is enough to process spatial relationships between floors, because the object of *partition* knows what floor it is in.

4.2 R-Tree indexing schema

The spatial data of indoor spaces contains information that is multidimensional and the objects like rooms are not well represented by point locations. But at the same time there is a great need to quickly and efficiently answer questions that require this information and evaluate it, f.x. distance between objects. In order to efficiently retrieve and manipulate objects of spatial data indexing mechanism is needed, that could be able to operate on spatial data.

The R-tree is a height-balanced tree structure created for multidimensional objects like polygons. The idea of the schema is to group objects that are nearby and represent them as an limited box called Minimal Bounding Rectangle (MBR). Every minimal bounding box is a part of hierarchy and because of this reason at every time moment only limited amount of data need to be accessed. It reduces the number of objects that need to be retrieved to operate with.

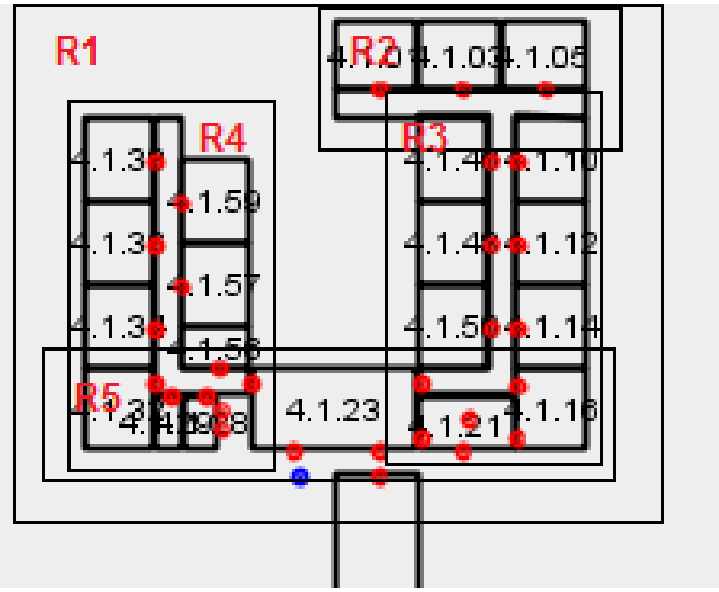

Figure 4.2: *The example of R-Tree indexing schema*

The example of R-tree indexing schema is given above in figure 4.2. The example used the data that was used for testing how queries work. In the example shows how MBRs are formed based on floor map. The bounding box $R1$ is the top layer of hierarchy it contains the partitions of one single

floor. The partitions $R2$ - $R5$ are inside this partition. In order to illustrate the retrieval process, we are giving the following example:

The partition named "1.2.1" belongs to MBR $R5$ and $R3$, and in order to look for the neighbours from those partitions will be looked through. In this way all the partitions that do not intersect the bounding rectangle also cannot intersect any of the contained objects. The number of processed objects are reduced.

The shortest path can be extracted as a sequence of objects that share common properties (f.x. the rooms that share the door), and for this reason it is very useful to extract and operate only on the data that is spatially close.

The data of indoor spaces is also not changing or changes are very small and very rare, but even if it changes, index is completely dynamic and the data will be maintained to keep it atomic with no periodic reorganization.

R-trees are very suitable for large spatial datasets, because:

- All the geometries are generalized to rectangular bounding box
- Operations are simple
- Easy to find overlaps and distances

4.3 Queries

The spatial data of indoor spaces hides answers to interesting questions, e.g. "What are the shortest paths from the room X to other rooms in a single floor buildings", or "What is the shortest path from the room X to room Y if X and Y are on different floors".

The combination of proposed data model and R-tree indexing structure enables getting answer to such questions. As it was mentioned in the sections 4.1 and 4.2, the designed data model with indexing schema is made for indoor spaces.

Even though the architectural objects of stairs and doors are very similar and both connects two partitions on the same or different floors, but the objects $access_point$ and $connector$ of our data model are separated from each other and therefore there are different queries to answer questions about single-floor or multi-floor buildings.

4.3.1 Single-floor buildings

The connection between rooms through doors is shaped as a non-deterministic graph, that might have cycles in it. It is important to understand that it could mean, that there is an unlimited number of paths possible to move from room X to room Y. And therefore we will be using recursive queries that removes paths, that contains partitions that have already been accessed in the current pathway.

The first query (4.1) returns all the possible pathes in a single floor. The path is considered to be shortest if it takes the least number of jumps in database. The query returns a number of parameters as a result, and the parameter 'depth' represents the number of the joins in a field named depth.

The algorith finds the solution by joining the partitions that are next to the first chosen partition and that has an actual access point to the other partition that has access to the same access point and not further than the *threshold*.

The partitions are noted as inner walls and therefore they do never overlap. To workaround this detail the parameter of threshold is introduced. It is calculated that 400 cm is the best distance in order to find only the partitions that are actual neighbours of a given partition.

```
1  WITH RECURSIVE search_graph(id, floor_id, data, depth, path
       , cycle) AS (
2      SELECT p.id, p.floor_id, p.geom, 1, ARRAY[p.id], false
3      FROM partition p
4    UNION ALL
5      SELECT p.id, p.floor_id, p.geom, sg.depth + 1, path ||
           p.id, p.id = ANY(path)
6      FROM partition p, door d, search_graph sg
7      WHERE sg.floor_id = p.floor_id AND ST_DWithin(sg.data,
           p.geom, $threshold$)
8        AND d.partition1 = p.id AND d.partition2 = sg.id AND
             NOT cycle
9  )
10 SELECT * FROM search_graph;
```

Listing 4.1: *The query for all the possible pathes in a single floor*

```
WITH RECURSIVE search_graph(id, floor_id, data, depth, path
    , cycle) AS (
    SELECT p.id, p.floor_id, p.geom, 1, ARRAY[p.id], false
    FROM partition p
    WHERE p.id = $from_partition_id$
  UNION ALL
    SELECT p.id, p.floor_id, p.geom, sg.depth + 1, path ||
        p.id, p.id = ANY(path)
    FROM partition p, door d, search_graph sg
    WHERE sg.floor_id = p.floor_id AND ST_DWithin(sg.data,
        p.geom, 400)
      AND d.partition1 = p.id AND d.partition2 = sg.id AND
          NOT cycle
)
SELECT * FROM search_graph WHERE cykle = false AND path @>
    ARRAY[$to_partition_id$];
```

Listing 4.2: *The query for the path for the shortest path between two specifically chosen locations*

The second query (4.2) is more specific. It returns the shortest path between two specifically chosen locations. It works on the same principle, except it checks the partition from where the search has to be started and filters all pathes that do not include given destination.

Both queries are recursive and therefore it is very important to forbid the loops of infinitive cycles. This is done by setting the cycle parameter to false.

4.3.2 Multi-floor buildings

For the buildings that contain several floors the query is a bit more complex and comlicated. The query, given in listing 4.3 checks for both connections between adjacent partitions sharing doors with current partition and also partitions on different floors, that have a connector that would connect them.

```
WITH RECURSIVE search_graph(id, floor_id, data, depth, path
    , cycle) AS (
    SELECT p.id, p.floor_id, p.geom, 1, ARRAY[p.id], false
    FROM partition p
        WHERE p.name = '016'
  UNION ALL
        SELECT p.id, p.floor_id, p.geom, sg.depth + 1, path
            || p.id, p.id = ANY(path)
        FROM connector c, search_graph sg, partition p
        WHERE ((p.id = c.partition1 OR p.id = c.partition2)
    AND (sg.id = c.partition1 OR sg.id = c.partition2) AND
       NOT cycle)
    OR (sg.floor_id = p.floor_id AND ST_DWithin(sg.data, p.
       geom, 400)
    AND d.partition1 = p.id AND d.partition2 = sg.id AND
       NOT cycle)
)
SELECT *
FROM search_graph
WHERE cycle = false
```

Listing 4.3: *The query for the shortest path between the partitions that are on different floors*

4.4 Distance calculation

The spaces that indoor navigations is useful for usually consists of quite large areas, which are not empty and easy to move through. The aisles of shelves or other objects are blocking the way, so that people can not pass the room straight away. And since it is not always possible to take the straight path, the Manhattan path is considered to be able to help. For a short reminder the definition of Manhattan's distance is given in definition 4.4.1.

Definition 4.4.1. Manhattan's distance is the distance between two points in a grid based on strictly horizontal and/or vertical path. The Manhattan distance is the simple sum of the horizontal and vertical components.

The partitions are in the shape of various polygons. For most of the time - rectangles, and sometimes concave polygons. But since the partitions are usually not empty and people are blocked from taking the shortest

straight ahead path, the calculation of distance is possible by calculating the perimeter between doors in the room which is equal to Manhattan's distance.

The example was given in the section 3.1.

However, the heights of the buildings are not being stored in the database and the complexity of calculation of distance on *connector* objects it is impossible to calculate the distances between several floors.

Chapter 5

Results

This chapter is dedicated for the settings of the experiments and results of those experiments.

5.1 Settings of experiments

This section serves as introduction to the results of experiments. The open source DBMS PostgreSQL was used for data model creation, queries and experiments. The data that was used for experiments was extracted from a number of diferent IFC files during previous semester project work and moved to newly created and adapted model.

5.2 Navigation queries

Two kinds of navigation queries were introduced in the project chapter: single-floor queries and multi-floor queries.

5.2.1 Single-floor queries

The visual result of single-floor navigation query (listing 5.1) is given in figure 5.2 and 5.1. The shortest path from the room 101 to the room 107 (the room id in the database is 52, and therefore it is 52 in the last line of query).

	id integer	floor_id integer	data geometry(Polygon,4326)	depth integer	path integer[]	cycle boolean
1	52	1	0103000020E6100000010	5	{65,44,35,42,52}	f

Figure 5.1: *The result in the database*

The shortest path is extracted as a sequence of adjacent partitions starting from the room number 101 (id = 65) and then consequently joining other partitions that are adjacent neighbours to already connected partitions and shares the same access point until the final destination is accessed.

The search on the database takes only 35ms when the level of query depth is 5.

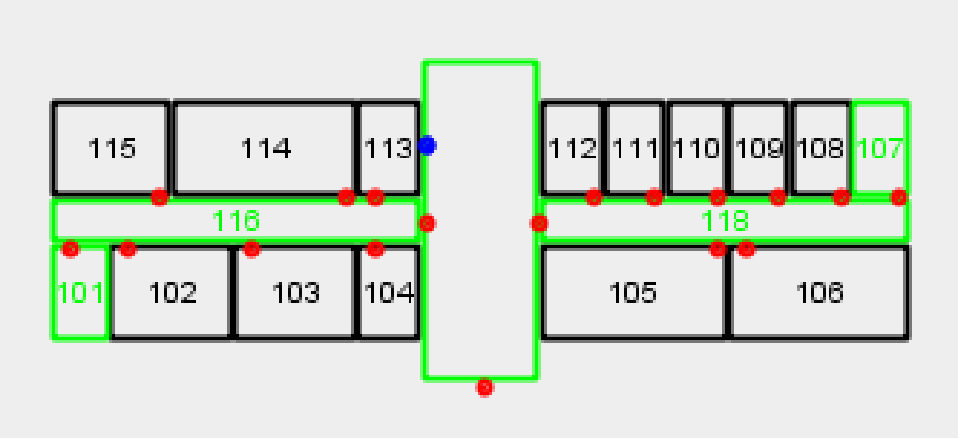

Figure 5.2: *The shortest path query visualization of query given above.*

```
WITH RECURSIVE search_graph(id, floor_id, data, depth, path
    , cycle) AS (
    SELECT p.id, p.floor_id, p.geom, 1, ARRAY[p.id], false
    FROM partition p
    WHERE p.name = '101'
  UNION ALL
    SELECT p.id, p.floor_id, p.geom, sg.depth + 1, path ||
       p.id, p.id = ANY(path)
    FROM partition p, door d, search_graph sg
    WHERE sg.floor_id = p.floor_id AND ST_DWithin(sg.data,
       p.geom, 400)
      AND d.partition1 = p.id AND d.partition2 = sg.id AND
         NOT cycle
)
SELECT * FROM search_graph WHERE cycle = false AND path @>
    ARRAY[52];
```

Listing 5.1: *The query for a single-floor building*

5.2.2 Multi-floor queries

The shortest path query for multi-floor environments is given below. It is a little bit more complex than single-floor environment query, because the connector objects are stored in different database table and they have to be taken into account only when it is needed.

```
1  WITH RECURSIVE search_graph(id, floor_id, data, depth, path
       , cycle) AS (
2      SELECT p.id, p.floor_id, p.geom, 1, ARRAY[p.id], false
3      FROM partition p
4      where p.id = 65
5    UNION ALL
6          SELECT p.id, p.floor_id, p.geom, sg.depth + 1, path
               || p.id, p.id = ANY(path)
7          FROM search_graph sg, partition p
8          WHERE (sg.id, p.id) IN (SELECT d.partition1, d.
              partition2 FROM door d UNION SELECT c.partition1
              , c.partition2 FROM connector c) AND NOT CYCLE
9  )
10 SELECT *
11 FROM search_graph
12 WHERE cycle = false and path @> ARRAY[16]
```

The query visualization is given below. Doors are marked with blue circles and connectors (stairs or elevators) with red ones. The query on different principle than the single floor query. The nested query on line 8 returns a list of all the pairs of connected partitions: not only of doors but also the conectors. And in this list the search works to find partitions that are on the accessible list and add them to the path.

The example query show the path between partition '103' to partition '304'. The results were given to show, that it is possible to achieve efficiency with such navigational queries. However, due to some missing relations between partitions it is not possible to find the path to/from those incomplete partitions.

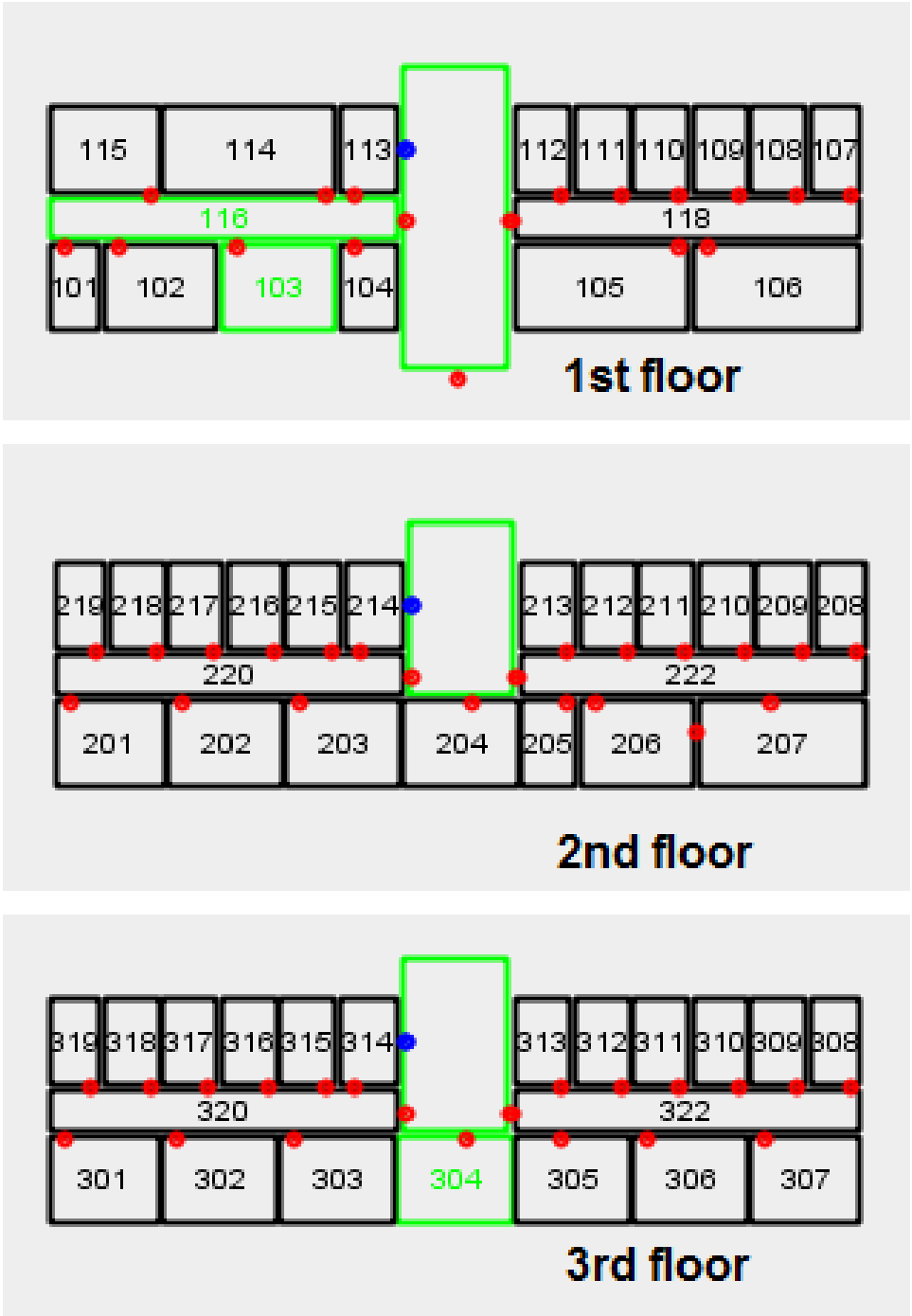

Figure 5.3: *The visual explanation of*

Chapter 6

Conclusions

As it is stated in the project definition, in chapter **??**, the problem of this project is to enable the shortest path queries in middleware-free environment. When mobile devices become smaller and users are getting less patient, the systems that serves customers must be fast and efficient

The solution to this problem is a new aproach to indoor navigation queries. The approach is different from others because it counts on complex sql queries instead of middleware environment.

As the solution, system contributed with following:

- The model for indoor space objects extended with R-Tree indexing schema
- The queries that can answer navigational questions related to shortest paths

The data model stores data not only about connections between elements and constraints of space, but also the geometrical data that is used to visualize software applications. The usage of those spatial elements is effective even with large spaces. In addition, the spatial properties of geometrical information can be used for more diferent purposes, e.g. distance calculation. The usage of dynamic and self maintaining R-tree indexing schema reduces the number of elements to search through and makes it easy to change data.

In the conclusion, the solution makes it possible to ask direct database queries to get answers to the questions about shortest paths in single-floor and multi-floors environments.

Chapter 7

Bibliography

[1] S. L. A and J. L. A. Efficient topological data models for spatial queries in 3d gis, 2010.

[2] I. Afyouni, C. Ray, and C. Claramunt. Spatial models for context-aware indoor navigation systems: A survey. *J. Spatial Information Science*, 4(1):85–123, 2012.

[3] M. B. Christian de Haas and A. Pilvinytė. Indoor space data: Prototype for extraction and implementation. Technical report, 2012.

[4] A. Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, June 1984.

[5] R. K. Harle. Spatial indexing for location-aware systems. In *Proceedings of the 2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking&Services (MobiQuitous)*, MOBIQUITOUS '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[6] S. H. Hussein, H. Lu, and T. B. Pedersen. Towards a unified model of outdoor and indoor spaces. In *SIGSPATIAL/GIS*, pages 522–525, 2012.

[7] K.-J. Li. Indoor space: A new notion of space. In *Proceedings of the 8th International Symposium on Web and Wireless Geographical Information Systems*, W2GIS '08, pages 1–3, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] M. F. Mokbel and J. J. Levandoski. Toward context and preference-aware location-based services, 2009.

[9] D. Papadias, Y. Theodoridis, T. Sellis, and M. J. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with r-trees. pages 92–103, 1995.

[10] E.-P. Stoffel, B. Lorenz, and H. J. Ohlbach. Towards a semantic spatial model for pedestrian indoor navigation. In *ER Workshops*, pages 328–337, 2007.

[11] P. yves GilliÃ©ron, D. BÃ¼chel, I. Spassov, and B. Merminod. Indoor navigation performance analysis. In *In Proceedings of the European Navigation Conference GNSS 2004*, 2004.