Authorized access to dynamic spatial-temporal data using the Truman Model

Master Thesis
of
B.J. van Velden [bart@vanvelden.org]

October 2007

Supervisors:
Dr. J.B. Voorbij
Dr. L. Breure

Study: Information Science
Education programme: Content and Knowledge Engineering

Department of Information and Computing Sciences, Utrecht University

INF/SCR-07-17
Abstract

This thesis presents a scenario in which the access to a dynamic collection of objects with both a spatial and temporal reference, has to be authorized to a large group of users. The authorization parameters are based on the spatial and temporal dimensions of the collection. After defining the requirements in a more formal way, existing authorization models are evaluated and found unsuitable for the projected scenario. Even the GeoXACML architecture, which is currently proposed as a new standard by the Open Geospatial Consortium, is found problematic.

Since mainstream databases have implemented spatial data types and spatial functions it seems feasible to enforce the authorization policies on the database level. The current SQL authorization mechanism is however restricted to the level of tables, views and columns. The traditional approach of implementing fine-grained authorization in the application layer also has several major drawbacks. The concept of the Truman model, in which authorization is performed by query modification on the database level, looks promising. The benefits of this model include a single point of authorization enforcement, the option of having a dynamic collection, end-user query abilities and efficient data processing (which is a major aspect with spatial data).

An implementation of the Truman Model using the Virtual Private Database feature of an Oracle DBMS combined with a proper data model for the spatial and temporal authorization conditions shows promising results for a collection of nearly 10 million geo-referenced 360 degrees panoramic images taken in a 10 year period. The architecture is further demonstrated using a setup including a secure web service and a 3D GIS application. The web service hereby functions as middleware and employs the user credentials (entered in the GIS application) as parameters for the database connection. Its basic functions are various retrieval tasks, including queries in which spatial-temporal data clustering is performed. The web service delivers the authorized data in the KML format suitable for Google Earth. This visualization method is used in an evaluation to present the effectiveness of applied authorizations in an understandable way. Additional results are presented in terms of query execution times, further recommendations and future work. The successful outcomes of the research project contribute to the general issue of spatial-temporal data authorization.
We accept the reality of the world with which we are presented.

Ed Harris as Christof in The Truman Show, 1998

Acknowledgements

With a solution to spatial-temporal authorization of information one can gain control of other people’s knowledge. This can result in opportunities, as well as risks. Therefore as always: ‘use it well and wisely.

There are many people I have to thank for supporting me in completing this research project. First of all I have to thank my supervisors at the Department of Information and Computing Sciences, Hans Voorbij and Leen Breure for their critique, encouragement and patience. I want to thank CycloMedia International B.V. for providing a suitable case scenario for the main topic of this thesis, and the provided resources. I especially want to thank Bernard Rutgrink, Leon Wolters and Sander Jongeleen for their time and useful feedback. Marieke, my sister, deserves some special attention since she kindly offered to review my thesis, which provided very useful comments to improve its quality. Last but not least I thank all close friends and family whose love, joy, and support enabled me to complete this work.

Bart van Velden

October 2007

Utrecht, The Netherlands
## Contents

1. **Introduction** .......................................................................................................................................... 3  
   1.1. Context ........................................................................................................................................ 3  
   1.2. Case scenario ............................................................................................................................... 5  
   1.3. Research question ........................................................................................................................ 6  

2. **Related work** ......................................................................................................................................... 7  
   2.1. Spatial-temporal database models ............................................................................................... 7  
   2.2. Authorization models .................................................................................................................. 8  
   2.3. Authorization evaluation on the database level ......................................................................... 10  
   2.4. The Truman Model .................................................................................................................... 11  
   2.5. Limitations of the Truman Model ............................................................................................. 11  

3. **Methodology** ........................................................................................................................................ 13  
   3.1. Implementation .......................................................................................................................... 13  
   3.2. Evaluation .................................................................................................................................. 13  

4. **Implementation** ................................................................................................................................... 15  
   4.1. Architecture ............................................................................................................................... 15  
   4.2. Tolerance ................................................................................................................................... 16  
   4.3. Physical datamodel .................................................................................................................... 17  
   4.4. User view ................................................................................................................................... 18  
   4.5. Logon trigger ............................................................................................................................. 18  
   4.6. Policies ...................................................................................................................................... 19  
   4.7. Architecture overview ............................................................................................................... 22  
   4.8. Authorization management ....................................................................................................... 22  
   4.9. Architecture utilization .............................................................................................................. 23  

5. **Evaluation** ........................................................................................................................................... 25  
   5.1. Authorizations management ...................................................................................................... 25  
   5.2. Visualizing authorizations .......................................................................................................... 25  
   5.3. Executing retrieval tasks ............................................................................................................ 27  
   5.4. Performance ................................................................................................................................ 32  
   5.5. Analysis ..................................................................................................................................... 34  

6. **Conclusion** ........................................................................................................................................... 36  

7. **Discussion** ........................................................................................................................................... 37  

8. **References** ........................................................................................................................................... 39  

Appendix A - In-depth Case Collection Description ................................................................................. 41  
Appendix B - Use Case Model .................................................................................................................. 45  
Appendix C - Model criteria ................................................................................................................... 46
Appendix D - Sample SFS geometries in Oracle ................................................................. 47
Appendix E - Spatial-Temporal Query ............................................................................. 48
Appendix F - Example spatial-temporal task- Example spatial-temporal task................. 50
1. Introduction
This chapter provides a general introduction to the research project. It starts by providing background information on the main topic of this research project: the authorization of spatial-temporal data. This will also illustrate the motivation for carrying out a research project on this topic. The second paragraph presents the research question addressed. The third paragraph describes a case scenario that is used throughout the entire project.

1.1. Context

Spatial-temporal data
There are many potential application domains for spatial-temporal systems, including environmental change monitoring, transportation, socio-economic and demographic applications, health and epidemiology, multimedia, governance and administration, crisis management, and defense. Additionally, the increased use of real-time, mobile and on-location sensors is leading to many new potential applications. The use of spatial-temporal information has also become the key data for enabling decision making in many business sectors [Galton and Worboys, 2005] [Matheus, 2005].

Chen et al. have carried out research on image browsing. They conclude that the main factors in episodic memory, being time and location, are useful for improving personal image management. A Time – Location Clustering Model is developed to create sets of related images. The existence of a large time or large location gap is used as a basis to separate images from each other. Similarly, both a small time gap and a small location gap, suggest that the images belong to the same event, and are thus related [Chen et al., 2006].

Bierig et al. have created a context model on three contextual attributes – time, location and interest – and used this to personalize search results of users in a mobile environment (e.g. as in location-based services). The impact on users' perception of usefulness caused by both time, location and interest was considerable. This research does not only confirm the validity of the chosen attributes as part of the context model, but also accentuates the importance of location and time references attached to information objects [Bierig and Göker, 2006].

Galton et al. state that although geographic phenomena not only have static but also dynamic characteristics, traditional spatial information systems hold only a single state of the real world. They further discuss that Geographic Information Systems are now (2005) beginning to offer some practical temporal functionality, but that truly spatio-temporal information systems are still in the research arena [Galton and Worboys, 2005]. This topic is endorsed by the initiators of the TimeMap project who came across the lack of off-the-shelf GIS solutions to handle time, and are since developing their own solution [Johnson, 2004].

Kraak has carried out a lot of research on visualizing spatial-temporal data. Most of it is based on the space-time model introduced by Hägerstrand at the end of the sixties. This model is a three dimensional presentation of space and time in a single image, in which the spatial dimension is projected on the basis (X-axis and Y-axis) and the temporal dimension on the Z-axis. With this model, one is able to represent spatial processes and events with a time reference. Examples are a Space-Time-Path, and a Space-Time-Prism. An illustration of the space-time cube is provided in Figure 1 [Kraak and Koussoulakou, 2005].
Data mining and knowledge discovery have become popular fields of research. A significant subset of this research is looking at the particular semantics of space and time and how they can be sensibly accommodated into data mining/retrieval algorithms [Roddick et al., 2004].

We can generalize the above statements by stating that space and time are important components in various information retrieval tasks.

**Authorization**

A collection containing spatial-temporal data can be very valuable. This may not only be the case because of its usefulness, but also because of high production costs or privacy issues. For this reason it is important that the data in the collection is only accessible to the right users. Since the data may however be used by a large number of users, having quite different purposes with the data, this is not a straightforward task.

Roddick et al. warn for potential privacy and legal issues of location-area services. They note that the abuse of data collected about individuals is creating alarm in some areas and that the inappropriate use of personal data can cause adverse legislative reactions. This may not only block illegitimate uses of the data but also potentially useful application. The investigation of legislative frameworks is therefore identified as a useful area of research [Roddick et al., 2004].

According to Matheus the protection of intellectual property and copyright issues has become an important aspect since information is available in digital format. This is because the digital content can be copied without quality loss and with a reasonable effort of time, equipment and money. After copying, it can be distributed using the Internet, again with little effort of time and money [Matheus, 2005].

Most applications in the area of spatial-temporal analysis require flexible fine-grained access control to the data. Several access control models have been proposed for conventional systems. However, these models are not adequate for geographical databases, because of the peculiarities of geographic data. Despite the importance of the protection of this kind of data, no efforts have been devoted to the investigation of suitable access control models and systems [Belussi et al., 2004].

The terminology used in the field of computer security is a bit vague. To prevent any misunderstanding, I will provide a description of some terms.

- **Access control** includes identification, authentication, authorization and accountability.
- In access control models, the entities that perform actions in the system are called subjects, the entities representing resources to which access may need to be controlled are called objects.
- Identification is how an entity tells a system who he or she is, and uniquely identifies a subject.
• Authentication is the process of verifying an entity’s claimed identity.
• Authorization is the process of determining which objects a subject is allowed to access.
• Permissions determine what a subject can do with authorized objects.
• Accountability is the process of associating a subject with its actions.

1.2. Case scenario
Roddick et al. provided several recommendations for research on spatial, temporal and spatial-temporal databases. In this very useful article several characteristics are defined that the research should generally include. One of this is “Use large, real databases – synthetic databases are often of less use in this area than in many others” [Roddick et al., 2004]. This chapter will therefore present a real scenario including a spatial-temporal collection and an authorization challenge.

CycloMedia Technology
CycloMedia Technology is a company specialized in the large-scale and systematic visualization of environments using 360° panoramic images, also called cycloramas. An example of a cyclorama is presented in Figure 2.

Figure 2 - An example of a Cyclorama, presented in two different projections.

The images are captured by a camera system on the roof of a car. Each Cyclorama contains the location and time of its production. The collection looks as follows:

<table>
<thead>
<tr>
<th>ImageId</th>
<th>RecordingLocation</th>
<th>RecordingDateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>43c5kf8h0cg</td>
<td>5.70698580234006,50.8474750817698</td>
<td>2005-03-12 15:22</td>
</tr>
<tr>
<td>6ju83ks7u6pg</td>
<td>4.59019800392403,52.4622636292842</td>
<td>2004-08-02 08:51</td>
</tr>
<tr>
<td>85s8hi4nl7u2</td>
<td>6.900811752263,52.4115443666161</td>
<td>2006-06-21 12:17</td>
</tr>
</tbody>
</table>

Table 1 - A sample of the collection in the CycloMedia Case

The ImageId presents a unique identification number which can be used to retrieve the data-file using a specialized viewing application. The RecordingLocation defines the location in a geodetic coordinate system with world-wide coverage: the WGS84 system. The RecordingDateTime includes both the date and time value. The current collection contains nearly 10 million cycloramas.

Cycloramas (the objects) in the collection are licensed to numerous clients (the subjects). A contract defines which objects a client is allowed to see. This definition normally includes both a spatial and a temporal component. An example of a spatial component may be: “The city-center of Amsterdam”, a temporal component may be “Images taken in 2005, 2006 and 2007”.

The current authorization architecture is based on the use of datasets. In such a dataset a reference exists for each single image in the collection that is authorized to the client. As will be explained in a later section, the use of datasets is very restrictive. CycloMedia is looking for a flexible solution for the authorization of the cyclorama collection. Since the company has ambitions to expand its working area to other countries, and a newly developed recording system has a much higher capacity, the collection is expected to grow.
rapidly. For this reason a dynamic, scalable and flexible solution is necessary. A more in-depth description of the company and the collection can be found in Appendix A.

**Case requirements**

Several requirements should be defined for the model to be developed. Those criteria will be used for evaluating related research and the quality of the chosen solution. The requirements follow from the data collection, the authorization task (using the spatial and temporal dimension), and the utilization of the collection, which can be defined as retrieval.

Since the utilization of spatial data is very widespread, it should be clear that the criteria will not cover all possible scenarios thinkable. In that sense it is also meant to define several global concepts and to enforce focus in the research project.

The requirements are modeled in two ways: a more formal definition of the applicable criteria and a Use Case Model. The criteria are divided into three parts: Collection, Retrieval and Authorization; and include the major requirements for the case scenario. Using the formal notation the requirements can be discussed and evaluated more precisely. The Use Case Model presents the main actors and their activities. By using several extensions and inclusions (which are standard components of a Use Case model), the criteria have been included as activities in the model as well. The model is especially useful to gain a quick overview of the complete requirements. To explain certain activities some notes have been added to the model.

The criteria, as approved by CycloMedia, can be found in Appendix C; the overall scenario model in Appendix B.

**1.3. Research question**

A literature introduction provided insight in the domain of authorizing spatial-temporal data. Several researchers explicitly state that no suitable solutions to this topic are currently available. A real-world case example shows that besides authorization, the utilization (or more specifically the retrieval) is an essential aspect.

Additionally, the case illustrates that the spatial-temporal collection can be dynamic, and that this aspect should be considered carefully. The research topic can be described more precisely as the authorization of a dynamic spatial-temporal collection, in which dynamic incorporates the objective of supporting the retrieval task. This leads to the following research question:

*How can authorization of spatial-temporal data in a dynamic collection be realized?*

Since the authorization is focused on supporting retrieval tasks, I will relate to this term throughout the project with this idea in mind. Since retrieval implies the reading/viewing access of an object, other permissions (which determine what a subject can do with authorized objects, e.g. what kind of access a user has to an object) will not be further discussed.

The next step is performing an in-depth analysis of relevant scientific literature. This will be presented in chapter 2. Based on the findings the methodology used for the further research will be chosen. At that point I will also elaborate further on the research question.
2. Related work

This chapter presents a survey and critical assessment of related work.

2.1. Spatial-temporal database models

Development and research in the area of spatial-temporal database models started decades ago, when the management and manipulation of data, relating to both spatial and temporal changes, was recognized as an indispensable assignment. However, due to the complexity of the data structures requiring careful analysis in structuring the dimensions, together with the representation and manipulation of the data involved, spatial-temporal data handling was not a straightforward task.

Pelekis et al. provide a literature review of spatial-temporal database models. It includes an overview of previous achievements within the domain and an evaluation of the various approaches through use cases and a comparison framework. One of most important norms of this framework is the ‘Type of change’ of the spatial-temporal object. The eight possible scenarios for changes are visualized in Figure 3 [Pelekis et al., 2004].

![Figure 3 - The eight types of change of a spatial-temporal object (Source: Pelekis).](image)

The CycloMedia case describes a collection which continuously grows. Existing objects in the collection are however static: Neither the object, nor the spatial or the temporal reference changes. The applicable change relating to each single object is thus ‘No Change’. What can be regarded as a change, is the relevant set of results for a given query. In the majority of the usage scenarios namely only the newest objects are relevant. The newest objects are however not simply the objects with the most recent temporal reference, since the mutual spatial distance between objects may be large. Both the spatial and temporal references are thus important in such a case. Additionally the user query could for example also contain a temporal range, thus excluding any subset of the collection.

Such complex selections from the collection, and the necessary inferences that have to be performed, are however not related to the objects itself, but rather to the context of the objects in the real world and the retrieval tasks of the users. It is therefore not important or relevant which ‘conceptual’ data model is chosen, but rather how the authorization task can be carried out on the collection, while offering users the ability to perform the complex retrieval tasks.

Traditionally, most Geographical Information Systems (GIS) can only query and display information that is under the control of the specific system (e.g. stored in a proprietary file format). Spatial data that are created and managed outside a specific GIS could not be queried in the GIS [van Oosterom et al., 2002]. This changed with the publication of Simple Features Specification for SQL (SFS) in 1999. This standard was set by the Open Geospatial Consortium (OGC), a non-profit, international, voluntary consensus standards organization that is leading the development of standards for geospatial and location based services. The implementation of SFS consists of an SQL extension that supports storage, retrieval, query and update of simple spatial features (points, lines and polygons) [OGC, 1999].

The well-known 9-intersection topology model [Egenhofer and Herring, 1991], recommended also in the SFS standard, shows the complexity of spatial relations between geometries. It utilizes the fundamental notion of topological primitives to investigate the interactions of spatial objects. The basic criterion to
distinguish between different relations is the detection of empty and non-empty intersections between the interior, exterior and boundary of objects. The number of detectable relations between two objects can be theoretically 512, but generally only a small part of them is relevant. Eight relations are given names, i.e. disjoint, meet, contains, covers, inside, covered by, equal and overlap. These relations are visualized for two polygon geometries in Figure 4 [Louwsma et al., 2005].

![Figure 4 – Examples of the main eight topological relationships between two regions (Source: Egenhofer).](image)

Mainstream DBMSs (Oracle, IBM DB2, Informix, Ingres, PostgreSQL) have recently implemented the spatial data types and spatial functions more or less similar to the SFS standard. This spatial functionality is typically available as an optional database extension. The implementations of spatial data types in the DBMSs are basically 2D [Stoter and van Oosterom, 2002].

### 2.2. Authorization models

Yu and Lim give a good classification and description of access control models: “Access control models can be discretionary or mandatory. In discretionary access control (DAC), owners of objects may grant access to others and are responsible for protecting the objects they own. In contrast, mandatory access control (MAC) assigns each object a security label that is used as the basis of restricting accesses of the users to the object.” DAC has been widely adopted by commercial applications and database systems. Due to its rather constrained way of granting access, the use of MAC has not been popular among commercial applications [Yu and Lim, 2004]. The use of DAC in relational databases is however limited to the level of a full table or view, and not to data rows [Chaudhuri et al., 2007].

Because it is only since recently that spatial relations can be evaluated in relational databases, another approach has been used for many years in the past. This cartographic approach is based on the use of a specialized tool for the evaluation of the spatial relation between objects and the authorization. Based on the result of this evaluation the objects are placed in static datasets for use in the GIS application [Belussi et al., 2004].

In 2004 Belussi et al. pointed out that the naive approach of building ad hoc datasets to support access control is not suitable when the user community is large and dynamic. Additionally, such an approach does not support flexible protection granularities and dynamic changes in the access control policies. The fact that the data to be authorized can also be dynamic is however not discussed. The authorization model proposed by Belussi relies on the classical discretionary model centered on the notion of authorization, which includes objects, subjects and privileges. An extra component, the window, is a spatial object which represents the region of space over which the authorization can be granted. A query language in algebra is proposed to formally define the authorization privileges and reasoning. The model however lacks a control mechanism that is able to enforce the proposed model. Although this part of research is planned as future work, no new results on this topic have been published [Belussi et al., 2004].

Since existing authorization models were inadequate to provide access control based on spatial and temporal attributes, Atluri and Chun proposed a Geospatial Data Authorization Framework (GSAM) in 2004. The framework is mainly targeted on the authorization of satellite imagery. In it, access control can be specified based on the region covered by an image, time of its capture and subject credentials. It is focused on user privileges such as view, zoom-in, download, overlay, identify and fly-by of raster data from satellite images [Atluri and Chun, 2004]. LTAM, a Location-Temporal Authorization Model was later proposed to include the spatial and temporal locations of mobile users in the specification of the
In 2007, GSAS (GeoSpatial Authorization System) was presented as the enforcement of GSAM. The system authorizes data by first selecting the authorized objects from the database, and then as a separate second step, retrieving those objects [Atluri and Chun, 2007]. This authorization verification process is visualized in Figure 5.

![Figure 5 - The authorization verification process of GSAS (Source: Atluri).](image)

Matheus focuses on Digital Rights Management for geographic information. His work is targeted on the Geography Markup Language (GML), which is an XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features. A major part of his paper is the description of geospatial access control, by the means of GeoXACML. This is a spatial extension of the eXensible Access Control Markup Language, abbreviated XACML. In a nutshell, XACML describes a general policy language that allows the formalization of access control restrictions using a rule-based notation. It defines XML Schemata for the structuring of policies and messages, encoded in XML. XACML limits the addressing (selection) of resources to elements of the resource content document, by the capabilities of XPath. In that sense, geographic resource objects – respectively geographic features – can not be selected, based on their geographic characteristics. This point has been discussed by several other researchers. Lu et al. conclude: ‘Although GML may utilize the readily available query languages developed for XML, these languages must be extended with spatial operators if they are to be used for GML’ [Lu et al., 2007]. Such spatial operators are based on the topological relations as described in Figure 4, and the result of an operator is always true or false. GeoXACML extends XACML by the definition of identifiers for the spatial types and spatial operator functions, which can next be used to define authorization rules for data with spatial features [Matheus, 2005].

To demonstrate GeoXACML, a prototype is presented. This prototype architecture consists of several software components, as visualized in Figure 6. In this architecture, the Authentication Service provides the functionality to handle a login challenge with the user. The Geospatial Policy Point (GeoPDP) can derive authorization decisions, based on geometry constraints. For each newly added identifier a function is implemented in the GeoPDP. The Web Map Service Policy Enforcement Point (WMS-PEP) is a farce to any OGC Web Map Service (WMS). It however analyzes the user parameters (such as GetMap and GetFeatureInfo) and creates a XACML compliant authorization decision request message, send to the GeoPDP. In order to do so, it first requests the user credentials from the Authorization Service. Based on the response from the GeoPDP (the authorization decision), the WMS-PEP either returns a message that the request is denied, or the requested map or feature information to the user. The WMS client is a regular Web Map Service client, but also provides the specific capabilities to handle exceptions that triggers a login challenge with the authentication service.
Since GML is set as the standard by the Open Geospatial Consortium to express geographical features and as an interchange format for geographic transactions on the Internet, GeoXACML has recently been proposed as a new standard. Currently a draft OpenGIS Implementation Specification is available, on which the OGC has announced a call for public comment [OGC, 2007].

GeoXACML is based on standards for spatial data (GML, WMS) and authorization (XACML), and provides a solution to enable access control to unprotected Web Map Services, without modifying the existing infrastructure. The prototype architecture, on which the concept is largely based, however shows some very inefficient consequences: all data is selected from the original database, is then converted to GML, and split and evaluated on a feature by feature basis against the authorization policies. Spatial indexes in the original database can thus not be used, and additionally the spatial comparison functions have to be implemented in the GeoDPD web service. Finally, the transportation of unauthorized data results in a higher load on processor and network capacity. In a scenario in which the original data can only be obtained from a WMS service, this solution might be useful, but in other cases one would look for another solution.

### 2.3. Authorization evaluation on the database level

The above analysis of available authorization models for spatial-temporal data showed various disadvantages. Most disadvantages stem from the fact that the authorization evaluation is performed outside the database. With the recent implementation of SFS in database systems it should however be possible to perform the authorization evaluation inside the database. Nevertheless no examples of such an approach can be found in literature.

To perform the authorization evaluation on the database level, a query has to be executed that results in the authorized set of objects for a user. Currently, the standard access control mechanism in SQL is coarse grained; in that it grants access to complete tables, columns or views. There is no direct way to specify fine-grained authorization to control which tuples are accessible to which users. Theoretically, it can be achieved by creating an access control list for each tuple. This approach is however not scalable, and would be totally impractical in systems with thousands or even millions of objects and users, as each would require an access control list to be provided by the administrator. Another approach one could think of is creating a view for every single user (Unlike ordinary tables in a relational database, a view is not part of the physical schema. It is a dynamic, virtual table computed or collated from the result set of a pre-compiled query.). Again this approach is not scalable with a large number of users [Rizvi et al., 2004] [Kabra et al., 2006].

Because of the above issues, current information systems typically bypass the database access control facilities, and embed access control in the application program used to access the database. This can be the end-user application or a middleware application. It is though problematic to embed the access control in
the end-user application, if one has no control over the application source, or if multiple applications are used. Changes in the authorization policies also have to be applied to multiple control mechanisms, but even then one depends on application update policies for the changes to become effective. Moreover the risk of users or hackers submitting arbitrary queries is present. A second option is to construct the query in a middleware application. This application should then edit queries executed by the end-user applications to incorporate the authorization policies. This can however become tricky in case of complex retrieval task and changing authorization policies. Another option is to provide several default retrieval tasks as functions to the end-user application. Although this would probably simplify the process, it directly limits the options for the users [Rizvi et al., 2004] [Kabra et al., 2006].

2.4. The Truman Model
For the above reasons, fine-grained access control should ideally both be specified and enforced at the database level. Rivzi et al. present the Truman model which is based on query modification on the database level to include authorization policies. Although the concept was discussed by others before, the Truman model generalizes the approach of query modification on the database level using a parameterized view framework. The idea behind the Truman security model is to provide each user with a personal and restricted view of the complete database. To realize this, user queries are modified to make sure that the user does not get to see anything more than is allowed. Effectively the authorization policies are added to posed queries as predicates, also known as logical expressions. This model is visualized in Figure 7.

![Figure 7 – The Truman Model in action: The query posed by the user is rewritten to query’ which is executed by the system.](image)

Since the modification of the query is done transparently to the user, the user may not even notice the existence of the access control mechanism. The architecture is thus perceived by the end-user as if no restrictions are applied to the data, and that he has access to all objects in the table. This view of the Truman model is shown in Figure 8.

![Figure 8 – The query modification under the Truman model is transparent to the user. For this reason the model can also be seen as one in which each relation in the user query is replaced by a view that the user is authorized to see.](image)

The Truman Model is sometimes also called fine grained access control or row level security. The name Truman Model is inspired by the artificial world spun around the character of Truman Burbank in the movie “The Truman Show” [Rizvi et al., 2004] [Kabra et al., 2006].

2.5. Limitations of the Truman Model
Although the Truman Model provides a solution to fine-grained and authorization-transparent access control, it suffers from various issues: efficiency, effectiveness, and validity. Although related, these issues will be discussed separately.

The issue of efficiency is related to the fact that because of added predicates, the rewritten query may be quite expensive to execute; and may have different, complex, execution characteristics from the query posed by the user. The user query and the authorization predicates may even contain redundant tests.
Removal of redundant test is an extra task for the query optimizer, and if not performed, the redundant tests would result in wasted execution time.

The second issue is a question of effectiveness of the access control mechanism. Kabra et al. describe two channels other than the query result, through which users can gain access to unauthorized information based on information leakage. These channels include exceptions/error messages and user-defined functions. To target this problem, techniques are described to distinguish an unsafe query execution plan and how to find an optimal safe plan [Kabra et al., 2006].

The third issue, validity, results from the fact that the query that is executed is a transparent modification of the user query. Because of this, the returned answer may be inconsistent with what the users expects to see. Rizvi et al. indicate this as the major drawback of the Truman model. A sample is presented in which a student calculating the average grade from a Grades relation, is provided with the average of his own grades, because the Grades relation in his query was replaced by a authorization view MyGrades. As a solution to this problem the Non-Truman model is defined, in which each user query is subjected to a validity test [Rizvi et al., 2004].

Kabra et al. mention the Non-Truman in their attempt to fix the efficiency and effectiveness issues: ‘Although the Non-Truman model is attractive for guaranteeing correctness, any Non-Truman implementation is likely to be unpredictable in the sense that a powerful query inferencing mechanism is required, and that inferencing procedures can never be complete’. Unfortunately this statement is not further explained, but I think it has to be interpreted as that no constructed set of inferencing procedures whatsoever will be able to cover the diversity of queries which can be posed by users. This causes a situation in which a query that is accepted by one database implementation, may be rejected by another. Because this unpredictability is highly undesirable, in practice the class of Truman models is preferred to be used [Kabra et al., 2006].

In my opinion the validity issue can be fixed much easier by the use of proper public synonyms to refer to tables with authorization restrictions. This way, users utilizing such relations are aware of the restrictions that follow from the authorization rules. Since the use of synonyms has been implemented into all major database systems, this idea should be relatively easy to implement.
3. Methodology

The popularity and importance of spatial-temporal information gained my interest as the research area of my thesis project. Especially the concept of geo-tagging digital images appealed to me. I therefore contacted a company that is the major player in this area: CycloMedia Technology. This party was immediately interested in participating. In an open brainstorm session on a more specific research topic, the authorization of the collection was quickly distinguished as a key issue for the company. An analysis of recent scientific literature revealed that the topic - authorization of spatial-temporal data - formed a scientific desideratum. This issue was thus chosen as the research topic. The final problem definition was defined as:

_How can authorization of spatial-temporal data in a dynamic collection be realized?_

Due to the dynamic nature of the collection, it is important that the authorization of the data is enforced right before the data is used. In information science the term retrieval is often used to describe the task of selecting data. The authorization mechanism should thus be applied immediately before the retrieval task is executed. The newest changes are otherwise not available to the user. An authorization mechanism such as the creation of static datasets, will therefore be found unsuitable. This requirements is not only found in the case scenario (complex retrieval tasks, collection growth), but also in literature.

An analysis of existing models showed various disadvantages. The main cause of those disadvantages is related to the architecture of the proposed solutions: the enforcement of the authorization is performed outside the database, hereby restricting the usage of the data. Another finding was that since most databases have recently implemented the Simple Features Specification, it should be possible to perform spatial authorization evaluations inside the database.

Since the current authorization mechanism in SQL is however coarse grained, authorizations can not be enforced by the database itself, let alone spatial or temporal ones. I therefore discussed authorization enforcement by mechanisms inside end-user or middleware applications. The composition of suitable query however turned out to be complex and the architecture was found unmanageable.

The Truman Model on the other hand looks promising. It is based on query modification and a parameter framework controlled by database itself, and provides each user with a personal and restricted view. Because the authorization restrictions are applied at query execution, it offers the ability to execute retrieval tasks and have a dynamic collection.

The combination of those two relatively new concepts (a spatial database extension and the Truman Model) poses as a solution to the main issue addressed in this research project. No examples of such this approach can be found in literature.

To decide to what extends the proposed architecture is indeed adequate, an implementation and a careful analysis are necessary. The availability of the CycloMedia case scenario is very useful for the a real test implementation. The analysis should be based on the collection, retrieval and authorization requirements defined for this scenario, and the premises and issues presented in the previous chapter. Besides that it should discuss the solution in a broader context.

3.1. Implementation

The CycloMedia case offers a real-world scenario to take the proposed theoretic concept a step further. The implementation should be based on a database system with spatial functionality and an authorization mechanism based on the Truman model. Another required component is a proper data model for the data collection and the authorization parameters. These authorization parameters, which may be of spatial or temporal type, define which data an individual user is allowed to access. The enforcement of those restrictions will be realized in the query manipulation process. The design and construction of the architecture will be described in detail in the next chapter.

3.2. Evaluation

Based on an evaluation of the test implementation it should be possible to conclude whether the proposed concept works in practice. The effectiveness of the applied authorizations should thus be proven.
Additionally the range of retrieval tasks (queries) that were defined in the requirements of the case description should be executed.

Analyzing spatial-temporal data in its raw format is quite hard, so using a visualization like the space-time cube is preferred. In such a visualization the spatial reference is however limited to the use of two dimensions. The spatial reference in the scenario includes three spatial dimensions (longitude, latitude and altitude), but there is no immediate need to authorize the data on the altitude dimension. In future this will probably be the first criteria to be added\(^1\). The space-time cube can thus be used as a visualization instrument. Belussi et al. made the interesting note that a spatial authorization contains a spatial entity and can also be modeled as such [Belussi et al., 2004]. Similarly the spatial-temporal authorizations can be visualized in the three-dimensional visualization.

Since three-dimensional GIS applications have recently come available, it is preferred to use such an application as a visualization tool. By providing both the authorizations and authorized data in this comprehensible way, it will be possible to confirm and present the validity of authorization decisions taken by the authorization mechanism.

Although the authorizations could be managed by executing SQL statements or a database administration tool, the use of a more user-friendly tool is preferred. The creation of such a specialized tool is however only a final challenge.

Besides the correctness of the authorization mechanism, retrieval time is another important aspect. This is related to the way users will commonly interact with spatial-temporal data: a GIS application providing a direct manipulation interface. User expectations of this interface style involve continuous representation of objects of interest, and rapid, reversible and incremental actions and feedback. The data thus has to be selected, retrieved and visualized within a certain time frame [Baeza-Yates and Ribeiro-Neto, 1999]. Since the time needed for the retrieval and visualization of the data are related to network and user-processor capacity, the main concern is the query execution time. Especially since both spatial data analysis and query modification are indicated as negative factors for retrieval performance, this is likely to cause some complications.

The results of the evaluation will be presented in the chapter following the implementation.

---

\(^1\) Authorization on the altitude level is important in case of cycloramas taken inside buildings, under bridges, on viaducts and by helicopters. This is however not a requirement for the current project.
4. Implementation

This chapter describes the implementation of the Truman model on a dynamic collection of spatial-temporal data, in which the authorization rules are based on the spatial and temporal dimensions of the collection. The implementation is based on the CycloMedia case scenario, and is thus related to the criteria defined in Appendix C. In the next chapter I will evaluate to what extent this implementation is adequate for such a scenario.

4.1. Architecture

For the implementation, a database management system with spatial functionality is necessary. Since most mainstream DBMSs (Oracle, IBM DB2, Informix, Ingres, PostgreSQL) have implemented the spatial data types and spatial functions more or less similar to the SFS standard, this is not really a problem [Stoter and van Oosterom, 2002]. The spatial functionality may however not be included in the standard version, but only as an additional component. Database systems with an implementation of the Truman Model are however scarce, only Oracle’s Virtual Private Database and Sybase row-level authorization are available [Chaudhuri et al., 2007].

Virtual Private Database (VPD) has several other names within the Oracle documentation, including row-level security (RLS) and fine-grained access control (FGAC). Regardless of the name, VPD security provides a whole new way to control access to the data. It is based on the idea of having a defined security policy function attached to a database table or view, which is executed each time data in the table or view is queried or altered. This function returns an additional piece of SQL - called a predicate - that is attached to the original SQL’s WHERE clause, before the SQL is used. It thus matches the concept of the Truman Model. The query modification is done in the query optimizer and is actually performed when the SQL is parsed and executed. When the SQL is executed it is actually the modified SQL that is executed on behalf of the user. This means that the policy function controls which rows of data are returned. The process can be thought of as a system trigger that is executed when a table is accessed that has a policy defined. An important characteristic is the dynamic nature of a VPD [Finnigan, 2003].

Sybase Inc. introduced row level security to the Sybase Adaptive Server Enterprise (ASE) Database in 2003 under the name of Policy-Based Access Control. It matches the Truman model on the notion of authorization by query modification. Just like Oracle the authorization rules are enforced through the combined capabilities of various components: Access Rules, Application Context and Login Triggers [Sybase, 2003]. Sybase is partnered with The Boeing Company, who licenses Spatial Query Server, to spatially enable their database product. Unlike with Oracle and the ASE database, this component and its documentation appear hard to obtain [Boeing, 2006].

In this project an Oracle Database Management System (Version 10g, Enterprise Edition) will be used. For the necessary spatial functionality the additional option Oracle Spatial will be used. The implementation of the Truman model will be based on the Virtual Private Database component that is included only in the Enterprise Edition.

The spatial functionality in the Oracle Database is basically 2D, although the new 11G version will also include 3D spatial data types and functions [Stoter and van Oosterom, 2002] [Ihm et al., 2007]. As explained before, the spatial data in the scenario does include three spatial dimensions (longitude, latitude and altitude), but there is no immediate need to authorize the data on the altitude dimension.

The policy function that define the predicates will include spatial and temporal conditions. For this, spatial and temporal values or ranges should be specified for each subject that indicate what data one is authorized to access. These values should also be stored in the database. As defined in the requirements, there will be sets of spatial and temporal ranges. This requires a suitable data model, which will be presented in the next paragraph.

The proposed architecture is presented in Figure 9. Three types of data are visible: Users, Authorized ranges and the Collection. These tables has to be created first. Two administrative tasks are defined: The first is setting up the Truman model using the Virtual Private Database component by creating predicates, policies, a login trigger. This task has to be carried out by a database administrator. The second task, defining the authorized ranges should be performed by an account manager; ideally using a suitable interface.
If set up properly, the authorization is next enforced by the database itself. After a user connects using his credentials, the login trigger stores a user reference in the session context and activates the policies. At the moment a query is posed by the user, the activated policies supply predicates to the query optimizer which modifies the query. The modified query is executed next. In it the collection will be used, and since the predicates include conditions containing a reference to the user’s authorized ranges, this data is also used. Finally, the result of the modified query is returned to the user.

Figure 9 - Architecture of the proposed solution: the authorization is enforced on the database level by the definition of a logon-trigger and a set of policies. This results in modification of a user query by adding predicates to the where clause. References to the authorized spatial and temporal ranges, which are also defined in the database, are included in the predicates.

The described architecture is build on a midrange laptop computer, with a dual core CPU and 2 GB of internal memory, which is evidently not an ideal platform. It is therefore important to discuss the performance and scalability of the proposed concept in the evaluation.

The architecture is setup using a database user called ‘bart’ which has been granted several specific privileges. Those include creating and editing packages, procedures, functions contexts, triggers and access to the row level security API. This username might be found in the following scripts when those objects are referenced to by another user.

4.2. Tolerance

Uncertainty is inherent in most spatio-temporal applications due to measurement / digitalization errors and missing or incomplete information. Assume for instance, a user with a PDA inquiring about the closest restaurant in terms of read network distance. Although the user may actually be on a road segment, due the inaccuracy of the GPS device, the system may fail to recognize this. Such a situation can be handled by defining a threshold or tolerance value to associate a level of precision with spatial data. The tolerance reflects the distance that two points can be apart and still be considered the same. The tolerance value must be a positive number greater than zero. For geodetic data (such as data identified by longitude and latitude coordinates), the tolerance value is a number of meters. For example, a tolerance value of 100 indicates a
tolerance of 100 meters. The smaller the tolerance value, the more precision is to be associated with the data [Oracle, 2007b] [Roddick et al., 2004].

In the CycloMedia case the recording location of a cyclorama produced by the present recording platform is known within meter accuracy [Verbree et al., 2004]. The new DCR7 system uses NovAtel’s SPAN system, which combines an integrated GPS receiver and IMU (Inertial Measurement Unit), offers an even higher precision of within a decimeter [NovAtel, 2007] [CycloMedia, 2007]. Additionally the defined spatial authorization ranges have a certain precision. A lot of polygon datasets available for free on the internet, have for example been resampled to a low precision level, such as 100 meter. This decreases the file size of such polygons, and reduces the execution time of spatial functions in database. For this project an accuracy level of one meter is chosen. This reflects the accuracy of the geographic data in the collection, and the necessary precision for the spatial authorizations.

### 4.3. Physical datamodel

Two kinds of data will be stored in the database, namely the collection data and data necessary for the authorizations. This paragraph shows how this is put in a data model that conforms to the criteria.

**Collection data**

The collection is relatively easy to construct in the database: a single table with columns for the image_id, recording_location and recording_datetime will fit. The image_id is designated as the primary key. As the recording_location and recording_datetime are used in the authorization predicates and retrieval tasks, indexes are created on both. The RecordingLocation is of type SDO_GEOMETRY. This is a spatial data type that can hold various kinds of spatial geometries. An introduction to this data type is provided in Appendix D. The data model for the table is visualized in Figure 10.

![Figure 10](image.png)

**Figure 10** – This image presents the data model for the collection. It contains a single table for the images which holds the data-object (which is actually a unique identifier to a datafile), a spatial reference, and a temporal reference.

Next the raw data is imported into the database. As the original data was in the Dutch National Grid (Rijksdriehoekstelsel in Dutch) format, a conversion was made to the World Geodetic System (WGS84). The collection finally contains nearly 10 million images made in the Netherlands in a time period of about 10 years.

**Authorization data**

The data model for the authorization ranges is a bit more complex. It is visualized in Figure 11. Each contract defines an agreed set of conditions between a client and the data-owner. The spatial and temporal conditions have been separated into two tables to have the ability to define multiple spatial, and multiple temporal ranges for a single contract. (E.g. a single area with 2 time ranges). A contract contains at least one spatial range and one temporal range, and belongs to a single client. A client can of course have multiple contracts. A separate table relates clients and database-users. This way, several users can be added to a single client. The spatial ranges will be put in the table spatial_conditions, the temporal ones in the table temporal_conditions. This naming was chosen for clarity reasons: each spatial or temporal range will eventually result in a condition.
Primary keys in the data model are labeled ‘PK’, whereas columns that should be indexed are labeled with ‘I’ and foreign keys are labeled ‘FK’. The db_user relates to database users. The model is designed to fit the requirements defined in Appendix C.

4.4. User view

Following from the requirements, three sets of data should be available for each user. They can be defined as:

1. A set of condition-sets containing a description and authorized ranges.
2. A set of authorized images: image_id, recording_location and recording_datetime.
3. A set of unauthorized images: recording_location and recording_datetime.

The first set offers a detailed view of the authorized ranges one has access to. Since a contract can contain both spatial and temporal ranges, each such combination in a contract should be presented. The second set contains all objects that a client’s user is allowed to see. The third set may reveal that more data is available in the collection than is currently accessible. In the CycloMedia case this might be used as an easy marketing instrument. The three datasets are also shown in Figure 12.

4.5. Logon trigger

In the Truman Model queries are modified to include data conditions. A part of such a condition is the range the user is authorized to. As the query modification is performed at query-execution time, the authorized ranges of the currently connected user should then be available. The condition_sets view will
contain this information. To restrict the ranges to those belonging to the current user, the tuples in this table should be restricted to those with a client_id that matches the client_id of the current database-user.

The client_id of the current database-user should therefore be available when this view is used. This can be realized by determining the value at the moment the user connects to the database, and storing it in the session context for later use. Since setting a value in the session context can only be performed by a procedure that is permitted to do so, both a procedure and a trigger are defined. The trigger is attached to the database logon event, and executes the procedure that finds the client_id and stores it. The procedure is put in a package named the_ctx:

```sql
CREATE OR REPLACE PACKAGE the_ctx AS
  PROCEDURE set_client;
END;
```

The code for the body of the procedure is:

```sql
CREATE OR REPLACE PACKAGE BODY the_ctx IS
  PROCEDURE set_client IS
    the_client_id VARCHAR2(100);
    BEGIN
      -- determine client_name
      SELECT client_id INTO the_client_id FROM client_users
      WHERE user_name = SYS_CONTEXT('userenv','session_user');
      -- store client_id in session
      dbms_session.SET_CONTEXT('THE_CTX','THE_CLIENT_ID',the_client_id);
    EXCEPTION
      WHEN no_data_found THEN
        dbms_session.SET_CONTEXT('THE_CTX','THE_CLIENT_ID','');
    END set_client;
END the_ctx;
```

The logon trigger is defined as:

```sql
CREATE TRIGGER bart.logon_set_authorization
  AFTER logon ON DATABASE
BEGIN
  the_ctx.set_client;
END;
```

As the trigger is attached to the main database logon event, it will be executed for each connecting user. Not all of the database users might however be present in the clients_user table. In such a case this would raise an exception error, and would prevent the user from logging in. This scenario is therefore covered by the no_data_found exception handler. The result of the above statements is that immediately after a user has connected to the database, the corresponding client_id is retrieved and stored for later use. The next paragraph will present how the value will be used.

### 4.6. Policies

This paragraph presents the policies that implement the spatial and temporal conditions on the objects-to-be-authorized, by defining SQL predicates.

**condition_sets table**

For the condition_sets a view is created as a join of the contracts, spatial_conditions, and temporal_conditions table. Each authorized combination of spatial and temporal conditions is thus reflected in this view.

The Truman Model does only describe the concept of attaching a predicate to a view or a table. A specification of how this should be done inside a real-world database system does not exist. Oracle chose to create a single package with all functionality related to the mechanism: functions are available to create,
view, edit and delete predicates to/from a table/view. What precisely happens in the database system during the process of query modification is not revealed, since the processes inside the query engine are hidden from the user. For now this aspect is not really important, since the main focus is on evaluating whether the main idea behind the Truman Model can be combined with SFS to realize the proposed goal. The closed nature of the query engine might eventually result in some practical problems, but for now one has to live with this fact.

The following predicate is defined for the $condition\_sets$ table:

\[
(WHERE) \quad client\_id = SYS\_CONTEXT(''THE\_CTX'', 'THE\_CLIENT\_ID'')
\]

This predicate prevents the user to see data that is not authorized to him, by removing all rows with a $client\_id$, other than the one which was set in the $the\_client\_id$ value in the session context ($the\_ctx$). This predicate has to be added to the $condition\_sets$ table. Oracle provides the function $dbms\_rls.add\_policy$ for this. It can however not directly add the predicate, but needs a function that returns the predicate. Such a function is called a policy function. I thus create a function $client\_id\_security$ in a package I named $exp\_security$:

```
CREATE OR REPLACE PACKAGE exp_security AS
FUNCTION client_id_security(owner VARCHAR2, objname VARCHAR2) RETURN VARCHAR2;
END exp_security;
```

The body of the function is:

```
CREATE OR REPLACE PACKAGE BODY exp_security IS
FUNCTION client_id_security(owner VARCHAR2, objname VARCHAR2) RETURN VARCHAR2 IS
    predicate VARCHAR2(2000);
    BEGIN
        predicate := 'CLIENT_ID = sys_context(''THE_CTX'', 'THE_CLIENT_ID'')';
    RETURN predicate;
    END client_id_security;
END;
```

Now that we have a policy function, the special function $dbms\_rls.add\_policy$ can be executed. This function attaches the policy function to a defined table or view. When data from the table is selected, the policy function is executed, and returns the predicate. This predicate is used to modify the query before it is executed. The first parameter of the $dbms\_rls.add\_policy$ function defines the user who owns the table (or view) which is defined as the second parameter. The third parameter gives this new policy a name, which might be used later to remove or alter it. The fourth and fifth parameters define which policy has to added, and where it can be found. The final parameter defines that the policy should be used only when data is selected.

```
CALL dbms_rls.add_policy(''BART'', 'condition_sets', 'condition_sets_policy', 'BART', 'exp_security.client_id_security', 'SELECT');
```

Now each time the view $bart.condition\_sets$ is queried, the predicate that restricts the rows to the current user is returned from the policy function $bart.exp\_security.client\_id\_security$. This predicate can then be used for query modification as described by in the Truman Model. In the following paragraphs this view will be used to authorize the actual data. In the evaluation will be demonstrated what the result of the query modification is for a user posing queries.

**Images_authorized and imagesUnauthorized tables**

The $images$ table should be authorized in two ways, by two different policies. The first policy should exclude all rows that do not qualify to the authorized ranges in $condition\_sets$. The second policy should exclude all rows of images that are authorized, and also hide the ImageId of all remaining.

Two options are available to realize this: the first option is the use of public synonyms, the second one is the use of views. The Oracle manual however states that a column-level policy, which is necessary for hiding the Imageld, cannot be applied to a synonym (section 15.10.3 of [Oracle, 2007a]). The only available option thus is the creation of two views for the $images$ table. Since the removal of the image_ids
for the unauthorized images can as well be done in the definition of the view, as opposed to a specific column-level policy, I chose this option.

The spatial evaluation should check for the recording_location to be inside the specified area (the geo column). Oracle Spatial provides the function SDO_INSIDE(geometry1, geometry2) for this. The first parameter specifies a geometry column in a table, while the second parameter specifies a geometry from a table or a transient instance of a geometry. This means that this function cannot be used in a query like:

```
SELECT *
FROM images
WHERE sdo_inside(recording_location, SELECT geo FROM condition_sets) = 'TRUE';
```

Instead this query should be rewritten to:

```
SELECT *
FROM images, condition_sets
WHERE sdo_inside(recording_location, geo) = 'TRUE';
```

This requirement thus makes it impossible to define a policy function which adds an authorization predicate to a spatial data table with the default spatial function. One could construct an alternative function that performs a check for each image on all spatial areas, but this would require extra work, and will likely result in decreased performance. Performing complex spatial evaluations in a predicate added by the Truman Model thus seems problematic.

I therefore had to choose an other option: instead of defining the views images_authorized and images_unauthorized as copies of the original images table, both are defined as a cross-join of the images table and the conditions_sets view. This way a predicate based on the default spatial function can be used. The SQL statement for the images_authorized should thus be:

```
CREATE VIEW images_authorized AS
SELECT *
FROM images, condition_sets
```

Also adding the temporal predicates the complete predicate for the images_authorized view that would be defined in a policy function would be:

```
(WHERE) recording_datetime >= start_date
AND recording_datetime <= end_date
AND SDO_INSIDE(recording_location, geo) = 'TRUE'
```

The solution however contains a hitch: in case an object from the collection meets the conditions for multiple condition-sets, it undesirably appears multiple times in the images_authorized view. To fix this problem the user has to use the distinct selector in each query, which is a real unsuitable way out. A similar problem exists for the fields coming from the condition_sets view, which are also unwanted. Another new view accessible to the user could fix this problem:

```
CREATE VIEW images_authorized_fixed AS
SELECT DISTINCT imageid, recording_datetime, recording_location
FROM images_authorized;
```

The alert reader would however have noticed that the predicates added to the images_authorized (and also the images_unauthorized) view do not include a reference to the session context, since this was already present in the predicate for the condition_sets view. Because of this the spatial and temporal predicates can also be included in the actual definition of the view! Additionally the fields from the condition_sets view can also be excluded. The resulting SQL statement thus:
CREATE VIEW images_authorized AS
SELECT images.*
FROM images,
authorized_sets
WHERE recording_datetime >= start_date
AND recording_datetime <= end_date
AND SDO_INSIDE(recording_location, geo) = 'TRUE';

Using this approach no spatial and temporal predicates have to be defined in policy functions. An analysis of both designs showed no difference in performance. This indicates that the manipulation of queries is not a heavy job for the query optimizer.

The above solution fits the projected purpose: the data is indeed authorized on the spatial and the temporal dimension. The Truman model is used to restrict the rows in the condition_sets view to those of the current user, and based on this personalized set of conditions, the actual data is authorized by joining tables. The new view that are created this way, hereby only contains the data that matches the personalized conditions from the condition_sets. The single query modification performed by the Truman model is thus the basis for the full authorization mechanism.

The images_unauthorized table is created in a similar way:

### 4.7. Architecture overview

The final architecture provides each user with three tables. One table contains the authorized ranges, one contains the accessible data, and one reveals what data is not accessible. The last table thus contains all data from the collection minus the data of the second table, and only describes the information objects by their spatial and temporal attributes.

Each of the tables is actually a public view based on a set of physical tables, which belong to a single database administrator, and to which normal users do not have direct access to. Although each view exists only once in the database, the contents vary for each user.

This personalized view is realized by performing query modification as defined by the Truman Model on the first table. This view normally contains all ranges, but due to the query modification the rows are restricted those that belong to the currently connected user. For this, a special function was used that supplies the query engine with a predicate each time the table is referenced in a query. This predicate includes a reference to a client identifier in the session context. A trigger responding to the login event of a user retrieves and stores this value. Using the client identifier, the query engine can restrict the rows in the first table to those of the currently connected user.

Unlike the first table, no policy functions are attached to the other two, and no query modification is performed. This is because doing so was found to be problematic (complex spatial/temporal predicates resulted in syntax issues), but also unnecessary. An existing policy function namely works for any reference to the table or view it is attached to. The definition of the data views can thus include the spatial and temporal conditions and reference the first table for the personalized authorization ranges.

In other words, a query modification policy is also applied in case the table or view is referenced in a sub-query, or when it is used in the definition a view. Utilizing this important fact wisely, shows that the Truman Model can be used to perform complex authorizations, which are otherwise impossible (e.g. directly in a query modification predicate). In the implementation this is demonstrated by authorizing data on the spatial and temporal dimensions. The combination of a single query modification policy and the creation of joined views shows the strength of Truman model.

Using the Truman Model on the basic level for filtering the user’s authorization ranges, and defining a view for the complex evaluation of spatial and temporal conditions thus seems a adequate solution. Before one can conclude if this setup is successful, the correctness and performance should first be evaluated.

### 4.8. Authorization management

To manage the authorizations a specialized tool has been developed. This tool corresponds to the data model described in the implementation phase and the use case model found in the requirements. I will not go into the details of the tool, but will mainly summarize its basic functionality:
• creating/altering/deleting/viewing of clients, users, sets, spatial ranges, temporal ranges
• viewing conditions_sets (combination of all authorized spatial/temporal ranges)
• import/export spatial areas with a 3D GIS application

The tool is developed to provide an easy way of testing the authorization mechanism during the evaluation phase. To have full access to the database, an account is used that has full access to the tables and views, so that it is not obstructed by any query modification performed by the Truman Model.

4.9. Architecture utilization

To utilize the architecture a software component has been developed. With the tool retrieved results can be presented in a understandable manner. It is based on use of the 3D GIS application Google Earth, which is able to retrieve data from a web server in the KML format (Keyhole Markup Language). Since no software component was available for interfacing between an Oracle database and KML, I decided to build such a web service myself. This also offered true flexibility for the needed functionality, which includes:

• Requesting user credentials upon login in the GIS application
• Connecting to the database using the supplied credentials
• Execution of retrieval tasks with spatial / temporal user preferences
• Visualizing results in a style like the space-time cube

The web service was build in the PHP programming language running on an Apache web server. References to the URLs of the web service and the various retrieval tasks it offers are provided to the GIS application in a separate KML-file, and are called Network Links. This file has to be added to the GIS application only once. In Figure 13 the availability of the Network Links in Google Earth is shown.

```
Figure 13 – Screenshot of the 3D GIS application with the Network Links on the left. Placing a mark in a checkbox activates a Network Link.
```
After a Network Link is activated, Google Earth connects to the web service using a URL. This URL also contains a reference to a certain retrieval task (a query) as a parameter. Other parameters specify the current viewport (the direct manipulation interface). The web server immediately responds to Google Earth with a pop-up screen asking for user credentials. The login screen is presented in Figure 14.

![Login Screen](image)

**Figure 14 – A popup requesting the user credentials.**

The web service uses the provided credentials to connect to the database. After a successful connection has been established, queries can be executed. The results are converted into the KML format by the web service and returned to the GIS application. The webservice and the query it poses do not include any data authorization mechanism, because this task is the sole responsibility of the Truman model inside database. Except for the above functionality, I consider the inner workings of the web service not further relevant in this context. In the evaluation phase this software component will be used to execute several retrieval tasks.
5. Evaluation

In this chapter the use of the Truman model for the authorization of dynamic spatial-temporal data will be evaluated. This evaluation will be based on the implementation presented in chapter 3 and the requirements as presented in Appendix C. Additionally it will refer to premises and issues presented in chapter 2 on related work. I will start by presenting how the authorizations can be managed, followed by an illustration of how data can be utilized. The results will be discussed in the final paragraph, in which the overall validity of the Truman Model based architecture will be discussed.

5.1. Authorizations management

Using the authorization tool described in paragraph 4.8, several clients, users and authorizations are added to the database. The spatial authorizations can be constructed from two resources: The GIS application and data from Statistics Netherlands (in Dutch: Centraal Bureau voor de Statistiek), which uses data from the Dutch Land Registry Office (in Dutch: Kadaster) [CBS, 2007]. Various combinations of single and/or multiple contract(s), spatial condition(s) and/or temporal condition(s) are defined. In the following chapters these authorizations are used to visualize the effectiveness of the architecture, and to examine the performance. An example of an authorization is presented in Figure 15.

5.2. Visualizing authorizations

A special retrieval task has been implemented in the web service to deliver the authorizations to the GIS application. This way, users can observe their authorizations along with the data. I will first focus on what happens due to the Truman Model in this task. After that, the results will be visualized using the GIS application.

Now suppose we activate this task using the network link in the GIS application. Instantly the GIS application contacts the webservice using the URL defined in the network link. The webservice quickly returns a message that it needs user credentials to continue. In the GIS application this results in a small popup window (as shown in Figure 14). We enter the user credentials of an existing client: ‘utrecht’, which was also found in Figure 15. After submitting, the webservice checks whether it can successfully connect to the database and moves on to the next step: submitting a query to retrieve the authorizations:

```
SELECT *
FROM   bart.condition_sets
```
This clearly is a simple query, which doesn’t include any type of authorization filtering, as we would normally expect from a client-application. Without the Truman Model the above query would be executed directly, and would result in all spatial and temporal conditions for all users. A security policy was however defined for the `condition_sets`. Because of this the query is transparently rewritten to:

```
SELECT * FROM bart.condition_sets
WHERE client_id = SYS_CONTEXT('THE_CTX','THE_CLIENT_ID')
```

We see that a predicate has now been added to the query. The predicate refers to a variable called `the_client_id`. This variable is known by the database: it was stored right after the webservice connected to the database by a login trigger. At the moment the query will be executed, which is immediately after it is modified, the variable is substituted with this value. This results in the following query:

```
SELECT * FROM bart.condition_sets
WHERE client_id = 20
```

This query filters the results on contracts of only the current client. These result-set is returned to the webservice, convert into KML and returned to the GIS application. In the following section we will see that indeed only rows belonging to the user are returned, and that the Truman Model successfully filtered results by modifying the query.

In Figure 16 the returned authorizations from the above query can be observed. We see that only the results of the client named utrecht, to which the current user belongs to, are showed in the GIS application. For each combined set of spatial and temporal conditions a set of polygons is created (start-date and end-date) by the webservice. These polygons are added on different altitude levels, which are related to the start-date and end-date. All authorized images fall in between a set of such ‘layers’. Since all such authorizations are listed in the GIS application, this task is also very useful for browsing and locating the authorized set of images.

![Image of GIS application](image-url)

**Figure 16 – An authorization presented in the 3D GIS application. The contract that was described in Figure 15 is showed in the lower section of this image. The image in the upper right shows the two layers for a time range.**
5.3. Executing retrieval tasks

In this paragraph several types of queries will be executed on the database. This will be accomplished using the web service and the GIS application. Two types of queries are very common with spatial data: Window queries and Nearest-Neighbor queries. Those were also described in the requirements of the CycloMedia case. Especially the Within-window query is very useful in determining the correctness of the authorization architecture. Additionally, a complex retrieval task, in which data is clustered based on the spatial and temporal dimensions, will be performed. The description of the tasks will include the SQL statements and will present results using the GIS application.

Window Query

Window or (within-window) queries are typical for a GIS interface in which a map is shown. The visible area of the map is called the window or bounding box, and is defined by geographic coordinates. A within-window query selects all geometry objects which have the ‘inside’ topological relationship with the specified geometry. In the setup of the architecture, the sdo_inside function was used for this task. Oracle however also provides a similar function that permits fast selections. It compares geometry approximations to reduce the computation complexity and is therefore considered a lower-cost filter. Because it only compares approximations, it returns a superset of the exact result set. This function should be used for within-window queries. The fast selection filter is called sdo_filter, and works similar to the sdo_inside function. The select statement to select all authorized images from the collection should therefore be:

```
SELECT imageid, recordingdate, recordinglocation
FROM bart.images_authorized
WHERE SDO_FILTER(recordinglocation, ?window ) = 'TRUE'
```

In this query, ?window should be substituted by a (square) geometry that defines the visible area in the GIS application. The complete constructor format is described in Appendix D.

Figure 17 – The results of the within-window query on the boundary of a spatial authorization.

In Figure 17 the result of a within-window query for the client from Figure 15 is shown. The user is authorized to the green area on the left hand side, combined with a broad time range. Each green marker

---

2 In case of a 3D GIS application, additional parameters such as camera tilt, camera distance, heading etc. can be useful.
represents an authorized object (a cyclorama) as a result of the query specified above. Cycloramas outside the authorized area are removed from the result-set by the database itself. As expected, all green markers are found inside the authorization polygons (and between the two layers representing the temporal range). The unauthorized objects are added to visualize the effectiveness of the authorization mechanism. To retrieve these unauthorized objects, a query similar to the one above was executed using the view `images_unauthorized`.

But why did the query on the `images_authorized` view deliver the correct results? Unlike the previous authorization task, no query modification was performed on this query. It did so because the `images_authorized` view is created with a definition that joined the `image` table with the `condition_sets` table. And because query modification is performed on the `condition_sets` table, even if it is referred to in a view definition, it will only contain authorized rows. The additional predicates in the `images_authorized` view definition further describe what relation between any image in the collection and the authorized conditions is allowed. The consequence is that each image currently in the collection is evaluated against the authorized conditions. The only thing the window-query does is select the subset of the result that is currently relevant. Figure 17 shows that this setup works perfectly, and confirms that the query modification performed by the Truman Model can successfully be extended to authorize data on more complex dimensions.

1) A spatial authorization: the neighborhood ‘Kralingen West’ in the city of Rotterdam.

2) With the temporal authorization range set to 1994-1998, both authorized and unauthorized images exist inside the polygon (visible by mix of green and red markers).

3) With a broader temporal range (1994-2008), all images inside the polygon are authorized to the user.

4) To show that all images in the spatial area are now authorized, only the un-authorized set is showed.

Figure 18 – Various screenshots of the GIS application are showed to present effectiveness of a temporal authorization range.
The consequence of a temporal authorization is visualized in Figure 18. The first image shows an authorized area. In the second image, both the authorized and un-authorized objects are retrieved when the temporal authorization range is small. Inside the area both authorized and unauthorized objects are presented. In third and fourth image, the temporal range is wider, so that all images inside the area are authorized. The variation in the results shows that the authorizations are successfully applied, and proofs that the Truman Model is suitable for authorization on the spatial and temporal dimension.

Just like the user specifies the spatial range he is currently interested in (by using the window), a time period could be used to restrict the returned objects on the temporal dimension. This is however not supported by this GIS application. Later on, a complex spatial-temporal query will be presented in which a user’s temporal preferences are indeed included. For now it is important to recognize that both the spatial and temporal conditions are effectively applied by the database system.

Figure 19 – An authorized cyclorama, depicted on the map with a blue marker, is opened in a window inside the GIS application. An un-authorized cyclorama, like the one with the text-balloon, can not be opened. A popup with a redirect to an order form, or any other sales channel could be offered instead.

Figure 19 demonstrates how the cycloramas are presented inside the GIS application. It also shows how the set of unauthorized data can be used.

**Nearest-Neighbor Query**

The goal of a nearest-neighbor query is to find the object in the collection that most closely matches a given description. Nearest-neighbor queries are relatively easy to understand and execute when the object description consists of only a geometry point. This spatial geometry might for example be selected from an address service or in a GIS interface. A mock-up of this is provided in Figure 20.

![Figure 20](image)

Figure 20 – The image on the left shows the user’s location of interest in a GIS application by a cross. The image on the right visualizes the nearest result. The relation between the two location offers two important descriptors: distance and direction.

Using spatial distance calculations the nearest object(s) is (are) easily found. However when the object is described by multiple dimensions, for example a spatial and a temporal one, it becomes more complex. Which object is for example the nearest: one with a spatial distance of 20 meters and a temporal distance of 10 days, or one with a spatial distance of 10 meters and a temporal distance of 20 days?
The type of objects and the user’s context are important in such discussions. In the CycloMedia case properties such as the direction, the surrounding of the objects, and the reason why new images were captured may be important. This thesis will therefore not go deeper into this subject, but will focus on finding the spatially nearest-neighbor only. More information on multidimensionality can be found in [Tao and Papadias, 2002] and [Kreveld et al., 2005]. In Oracle Spatial the spatially nearest-neighbor can be found using the sdo_nn function.

After executing this query the first results looked satisfying. A nearest image was found in most cases. However I stumbled upon the fact that at some locations no result was found. As no constraints (like a maximum distance) were defined, this was unexpected. An investigation using some test-data revealed that the nearest neighbor query may, as described in the manual, need to be evaluated multiple times in ‘order to return the desired number of results that also satisfy other conditions in the WHERE clause’. The WHERE clause in the above statement does not, however, contain any other conditions! After executing some queries on the test-data, I found out what happened: no result is returned, if an unauthorized object is found nearer than the nearest authorized image. The consequence is however that information is leaked about where unauthorized objects are located. To fix this, the sdo_nn function also needs to be evaluated multiple times in case the referenced table is created using a join that filters the data, as is the case in the image_authorized table. As a user might be unaware of this, this finding was surprising, but also means a problem for using the Truman Model for spatial data authorization.

Fixing the issue of the sdo_nn function with the optional parameter sdo_batch_size, which can be used to choose the initial number of returned rows likely to satisfy the constraints in the where clause, is of no use either. This is because the effects of the applied constraints are unknown: the spatial and temporal conditions might restrict the access to any number of objects. To summarize: it looks like the sdo_nn function is entirely based on the index, and not on actual table rows.

Again I will not go deeper into this subject, since finding the nearest neighbor can be the subject of an entire new research project. I will however present a solution to the problem, that in my belief returns the correct results in any case, but doesn’t fix the leakage issue of the sdo_nn function. This solution is the following SQL statement:

```sql
SELECT imageid, recordingdate, recordinglocation, distance
FROM (SELECT imageid, recordingdate, recordinglocation, 
sdo_geom.sdo_distance(recordinglocation, ?geometry, 0.05) distance
FROM bart.images_authorized
WHERE sdo_within_distance(recordinglocation, ?geometry, 'distance=25') = 'TRUE'
ORDER BY distance) nearest_authorized_images
WHERE rownum = 1
```

The ?geometry has to be replaced by an instance of a geometry, as described in Appendix D. This statement works as following: it first selects the objects within the authorized set that are within a certain range from the requested location. Secondly it calculates the precise distance to this location for each object. In a third step the results are ordered ascending by the distance. Finally it limits the returned rows to only the first one. The range is not only a way out, but is rather useful. This is because with cycloramas the distance is important: it is unlikely that a cyclorama taken far from the requested location is useful in any way. This query will thus return an empty result-set if no authorized cycloramas exist within the specified range. In Figure 21 the implementation of the nearest neighbor task is visualized. In the final version the cyclorama is opened in the direction of the target. For this I constructed a special function that calculates the angle between two (WGS48) geometry points in radians, since this was not available in the database.
Complex spatial-temporal Query

As described in the requirements, users are often interested in the newest cycloramas only. The task presented here will therefore select the newest set of results that cover a defined area.

The current approach to realize this is based on the recording strategy and the use of datasets: areas are photographed completely, put in a new dataset, and those replace any existing dataset. The downside of this approach is that when a new dataset is incomplete, a gap is created. This is of course undesired for a solution in situational awareness. Besides this issue, the management of the datasets requires a lot of work.

To accomplish this task in the new architecture a proper query has to be formulated. This is however a rather complex task, as it involves both spatial and temporal relations between objects: for each object the spatial distance to other older objects has to be analyzed, and when such an object is found, it should be excluded from the result-set. When the temporal distance is however small, it should not be excluded, since both objects are relevant. To understand this, one should realize that the cycloramas are recorded in a continuous sequence. Because of this, two cycloramas with a small spatial distance, are also likely to have a small temporal distance. A certain temporal threshold is thus important.

The final query for this task can be found in Appendix E. In the web server this task is implemented twice: the first version only retrieves the desired results, while the second version provides an explanation of the task inside the GIS application. A screenshot of the explanatory version can be found in Figure 22. It visualizes the three dimensional clustering method by drawing cylinders, and showing both the newest and expired objects. The expired objects are colored yellow.
Figure 22 – The area shown in the GIS application is covered with the most recent objects from the collection. The vertical dimension, normally the altitude, is used to visualize the temporal dimension in the collection. The highest objects are the newest. The green cylinder around each newest object represents the spatial-temporal area it covers. Objects inside such a cylinder (depicted with a yellow marker) could be removed from the result-set, since they are no longer relevant.

A detailed example of this task can be found in Appendix F. Supported with several images it shows the practical usage in relation to retrieving all data.

The current task is designed to select the newest set of images covering an area. The opposite of this, selecting the oldest set of images, might also be useful. Especially since the collection is more and more becoming an important source for cultural heritage, such selections might be considered useful by users. This stresses the importance of the ability to execute all kinds of retrieval tasks, as opposed to only having static datasets.

The successful result of this task shows the general strength of the architecture. A complex retrieval query can be posed to the database without considering data authorization at any moment in time. The query is modified automatically and transparently by the Truman Model on the database level. The complex result-set suffices both the complex selection conditions and the spatial and temporal authorizations. To what extend such results are also delivered within a useful time frame, can be found in the following paragraph.

5.4. Performance
Besides correct authorization decisions, the performance of the architecture is important. The testing platform on which the architecture is build is not ideal: both the Enterprise DBMS, the middleware software and the GIS application are running on the same machine. The main hardware components of this laptop computer are an Intel® Core™ Duo Processor T2400 (1.83 GHz, 667 MHz FSB), 2 GB of internal memory (667 MHz), and a single 60 GB hard drive (Serial ATA, 7200RPM, 8MB buffer). The operating system is Microsoft® Windows® XP Professional with Service Pack 2.

When constructing the setup, the coordinates were converted to a different coordinate system, and spatially indexed. This literally took the machine half a day for the collection of nearly 10 million objects. This was kind of frightening, and the first results afterwards were also a bit disappointing. Some index optimizations however, which again took quite some time, showed better results.

Retrieval time
The final retrieval performance of the test implementation is quite satisfactory. In most cases the results are presented in the GIS within seconds. Especially the task that retrieves all authorized objects in an area, and
the one that retrieves the nearest object are executed very fast. Query times for the authorized and unauthorized sets are comparable. Covering an area with the newest objects clearly takes more time, and can frustrate the entire system when quickly moving around in the GIS application. This problem could nevertheless be handled by the middleware application by cancelling a query upon request of a new set. In the nearest-neighbor task, the specialized function was not used. The alternative statement is however rather fast: in less than a second the nearest result is located from the authorized set of objects. The complex spatial-temporal query takes noticeably longer time to execute, but when zoomed in on a small area also delivers the data within seconds.

The described retrieval speed is perfectly suitable for use with a direct manipulation interface. Especially when zoomed in on a useful altitude level (between fifty meters and several hundreds of meters) the results are present almost directly. I therefore conclude that dynamically authorizing data on the spatial and temporal dimensions using the Truman Model is found successful in terms of performance.

**Query execution time**

The retrieval time described above, includes the request message from the GIS application to the web server, the query composition, query execution (including the transparent query-modification) and KML generation. The query execution time is of course the most important thing here. Several queries were therefore executed directly using a query tool. This quickly made me realize that the query time was influenced by quite a few variables. These possibly could be the number of contracts, the number of spatial and temporal conditions, the size and complexity of the spatial conditions, the width of the range in temporal conditions, the total amount of objects that intersect with the bounding-box of spatial conditions or the range of the temporal conditions, the size of the result-set, the query complexity, etcetera. Additionally the indexing method used will possibly also support some of the variables better than others.

It is however very hard to isolate the described variables. When for example the number of spatial conditions is doubled, the amount of spatial comparisons will also change. It is however unlikely that this value will also double, since the data collection is quite random on both the spatial and the temporal dimension. An added spatial condition will thus probably lead to a larger area, a larger candidate set, a different number of spatial comparisons, a different number of temporal comparisons, a different size of the result-set, etcetera. Moreover the values will be different for each location and time frame the user is interested in. Additionally the test platform is not designed to do such extensive testing. It is therefore not useful to provide precise query execution times, and I will therefore limit this subject to the how to overall performance of the architecture.

**Simulation of database growth**

The scenario describes a collection that continuously expands. Support for this is also an important requirement of the Truman Model. Just to remember: the static nature of datasets formed a significant restriction! The proposed architecture, based on authorization enforcement using queries, of course supports this requirement. I found it therefore unnecessary to invest time in creating some kind of continuous data insertion process. It is however interesting to see how the system performs with a much larger collection. I have tested this by generating random data, which was inserted to a new table with a copy of the collection. To keep the scenario real, the spatial reference of the new data is related to the existing collection: For each existing object which is not expired, a new object is inserted within a range of 3 meters (random distance/random direction). The temporal attribute is set to a random date within the last 10 year. The new table finally contains 16 million objects. The next step is to recreate the indexes and define the views for an authorized and unauthorized set.

To compare the performance of the larger collection with the real collection, several within-window queries are constructed and executed on the two collections using the query tool. The result is that the execution on the larger collection takes longer, but is very similar to the normal collection. Only a minor difference in the execution times is found. This shows that the performance will not immediately be affected by growth of the collection. The results for a query in which spatial-temporal clustering is performed might be different, since the number of related objects will likely increase exponentially. A high-end server should in my opinion nevertheless be able to handle such situations as well. Overall the architecture is perfectly suitable for the dynamic collection is was proposed for.
**Multiple concurrent users**

In a real implementation the database will continuously be queried by a large number of users. It is important to know how well the architecture is able to handle this. Although the current hardware platform is not really designed for this purpose, some testing has been carried out anyway. This test was setup as follows: four client computers were connected to the testing platform using a network. On each of the clients the 3D GIS application was installed and provided with a network link to the testing platform. Using the Account Management Tool several accounts and authorization conditions were added to the database. When the setup was ready, users were asked to simultaneously login in the GIS application using their individual credentials and perform some browsing activities on both the authorized and un-authorized sets of data. During the test the performance of both the clients and the ‘server’ was monitored. The result was that the server was able to deliver the data requested, but that a slight decrease in speed was noticeable. No hard statistics have been measured during the test, but the fact that the clients were able to continuously use their direct manipulation interfaces should indicate that the system performs quite well. As the Truman Model needs a separate database user for each user, it might become unmanageable when lots and lots of different users are added. I will discuss this topic further in the discussion.

**Overall performance**

At this point it is primarily important to conclude whether the architecture can in practice realize an appropriate overall performance, or that it clearly will not be able to do so. The answer is positive: the test implementation was able to suffice the requirement of using a direct manipulation interface. Some scalability testing also approved this. Performance is thus not a problem for the Truman Model, even with complex spatial and temporal authorization conditions.

Before the proposed architecture will be implemented for real, one needs to undertake some serious testing and assess the optimization possibilities. Options like using a database grid, data arrangement optimization (spatially sorted, optimal block size/tablesparse defragmentation, memory allocation, etcetera) and indexes optimization (partitioning, memory-usage, etcetera) for example, have not been discussed here, but need to be optimized to get the most out of the hardware. In the case of a real implementation of the proposed architecture, there are thus many ways to tune the performance. Additionally the new database release (Oracle Database 11g) might perform even better out of the box.

### 5.5. Analysis

In this final paragraph the capabilities and limitations found throughout the evaluation phase will be summarized.

In paragraph 5.3 the correctness of the test implementation was evaluated. By visualizing the results of several retrieval tasks in the GIS application, the correctness of the authorization mechanism was monitored. This shows that in general both spatial and temporal authorizations are correctly enforced by using the Truman Model. Using a special function to select the nearest neighbor however showed some information leakage. This might be considered a major problem for some applications, and affects the general confidence in the architecture. This is thus a good example of the point made by Kabra et al. on effectiveness issues of the Truman model [Kabra et al., 2006].

Rizvi et al. stated that the performance of the Truman Model could be problematic [Rizvi et al., 2004]. Especially with a large number of spatial evaluations this was expected to be a problem. Paragraph 5.4 showed that the test setup was however able to perform quite well. Some scalability testing also presented satisfying results. Given the presented possibilities to performance tuning and the continuous developments in processing capacity (both in hardware and software), I conclude that performance is not an issue for the combination of the Truman Model and SFS.

The ability of having a dynamic collection was not explicitly demonstrated in the evaluation, since it is implicit to an SQL solution. It is however important to mention here that the proposed architecture is capable of it, since this was a major requirement. Adding objects to the collection is thus no problem, and can be performed at any moment in time. The access to such newly added objects will immediately be handled by the authorization mechanism, and the objects will be instantly available for retrieval. Adding new objects will however impact the data indexes. It might therefore be wiser to insert new data in a large
batch as opposed to continuous insert operations. The performance test with a large set of random data illustrated that the authorization model is indeed able to handle a larger set of data.

Related to the dynamic nature is the support for utilization of the collection: with SQL users have the ability to define their own retrieval tasks. Several types of retrieval tasks were executed to demonstrate this feature with the webservice. But since the Truman Model will take care of the authorization inside the database, users can also be given direct access, and thus have complete freedom in constructing their own retrieval task.
6. Conclusion

In this thesis, a solution to the authorization of a large and dynamic collection of spatial-temporal data is proposed. The architecture is based on two new concepts for database management systems: the Truman model and SFS. The Truman model uses query modification to enforce access restrictions on data objects on the database level, whereas SFS adds support for geometries and spatial relationships.

A main advantage of the architecture is that the authorization mechanism doesn’t obstruct the usage of the data in any way. This was distinguished as a major problem with other approaches to the authorization issue. Using the Truman model, complex retrieval tasks can be executed on the database level. It can thus be expressed in the SQL syntax, make use of available operators, use data-indexes and complicated query optimization mechanisms, without considering authorization restrictions at any moment. Especially with spatial data, the ability to use the indexes and operators is of utmost importance.

Another important benefit of this architecture is that it is able to handle a dynamic data collection. This is because the Truman model works ‘on-the-fly’, by altering posed queries. Therefore, data can be added or modified to the collection without any further consideration. Because of this, data can be delivered to the users without any delay.

A third benefit of the architecture is that the common range of tools used for spatial data visualization and management can still be used without modification. These tools are desktop GIS applications and map engines. The desktop GIS applications are mainly used by power-users for data analysis, while the map engines render ‘geographic image maps’ for use on websites. Such applications are primarily designed to display data, and do include authorization mechanisms. Since the proposed architecture performs the authorization enforcement on the database level, the applications are in no way hindered and can thus still be used.

Proof of concept for the proposed architecture is provided by a successful test implementation of the architecture. This showed that it can be implemented as described, and that both spatial and temporal authorizations are enforced correctly. This is visualized by presenting both the authorized data and the spatial-temporal authorizations in a 3D GIS application. A variety of retrieval tasks, including within-window queries, nearest-neighbor queries and one in which data is clustered (based on the spatial and temporal dimensions), show how the architecture can be utilized.

Using the default function to find the nearest neighbor, some information leakage in the authorization mechanism was detected. An alternative query was however able to fix this issue.

This research proposed and confirmed that by utilizing two newly introduced concepts for database management system, namely the Truman model and SFS (spatial types and functions), dynamic collections can dynamically be authorized on the spatial dimension and temporal dimension by a DBMS itself, while providing support for the retrieval.
7. Discussion

Although the final architecture is quite capable in carrying out the authorization task, some issues have to be discussed. I will discuss those in the following paragraphs.

The information leakage issue
The sdo_nn function showed some information leakage. It namely revealed to a user if an unauthorized object was found nearer to a requested location than any authorized object. This should not happen. No further examples of this issue have been found, but might be present in other default functions. This is a setback to the general confidence in the Truman model. In the CycloMedia case the problem can however be fixed using a different query.

Architecture scalability
In a final implementation at CycloMedia it is rather unlikely that users will be given direct access to the database. This has several reasons: the issue of information leakage, resource performance protection, standardization, Oracle licensing limitations, etc. A setup such as the middleware application is more likely, since it offers a solution to all those problems. It is here also important to note that in such a setup the architecture can be altered, to provide some additional optimizations. This has to do with how the database is informed on who’s behalf a query is executed. In the test implementation this was realized by a login trigger. This requires each user to be added as a ‘database-user’, and also requires the middleware application to make a separate database connection for each individual user. If however only a small (and controlled!) number of middleware applications will be used, and no end-user access to the database is necessary, the middleware application can also inform the database on who’s behalf the next query is executed, by setting the application context along with that query, or within a transaction. A single persistent connection can then be used, which is of course much more efficient. This ‘One Big Application User’-Model (as Oracle refers to it) provides additional scalability potential, but makes the middleware application responsible for the additional task of user switching.

Authorization types
In the CycloMedia case it is currently sufficient to authorize the access to the level of entire cycloramas. In future it might also be useful to authorize the access within a cyclorama, so that users can only look in a certain direction. Another type of access control, which is actually implemented in the current authorization mechanism, is the total number of cycloramas that can be viewed by a user. This is realized by performing a check on the number of supplied cycloramas, against the defined limit for that user. The check is performed by a server-side software component that supplies decryption keys for the data. It should of course be practical to also implement the described features in the database server itself. The current situation is however that the data-files (containing the image data) are stored separately from the database, which makes an implementation of these features in the database almost impossible. This aspect should therefore be reconsidered when the proposed architecture is indeed implemented.

Besides the positive authorizations seen in this thesis, negative authorization can also be of use. In the CycloMedia case it might for example be useful to restrict the access to a certain private area to only a single client. Figure 23 provides a visualization in which data from a holiday resort is not visible under the town’s account.
Permission types

This research project has been focused on authorizing the ‘read’ permission on data objects, based on their spatial and temporal properties. Other types of permissions have not been discussed. To what extends the proposed architecture is also suitable here is thus unclear. Further research is required for such scenarios.

Oracle’s implementation of the Truman Model

The current implementation of the Truman Model in the Oracle database, is rather complex. Setting up the Virtual Private Database is cumbersome: Triggers, Functions, Policies and Views have to be defined, and are spread across several places. Not to mention that both functions and triggers also have to be put in packages. The result is in the end that the actions of the Virtual Private Database component are not only transparent to the user, but also to a database administrator. This gives the feeling that the VPD is more like a plug-in or workaround than a real part of the database design.

In an article recently published by Microsoft Research, the Truman model (which is not implemented in Microsoft’s database product SQL Server) is discussed. It presents a novel approach to assigning predicate grants by the design of a strict generalization of the current SQL authorization mechanism [Chaudhuri et al., 2007]. An example granting query is:

```sql
GRANT SELECT employees
WHERE emp_id = user_id()
TO PUBLIC
```

Predicates are thus included in a normal grant statement. This is opposed to the Virtual Private Database implementation in Oracle, which decouples the policy specification from the SQL grant model. In my opinion this new approach is the way fine-grained access control should be, and I would like to see this proposal to be the basis of a full reference implementation.
8. References


Appendix A - In-depth Case Collection Description
CycloMedia specializes in the large-scale and systematic visualization of environments using 360° panoramic images (cycloramas). Due to, by CycloMedia developed recording- and process technology, large areas are photographed and entered in an online database. From each recording location, orientation and time are registered, which makes versatile applications possible, such as 3D measurements and modeling.

CycloMedia has clients in diverse markets such as central and local government, homeland security and financial institutions. By using cycloramas clients are able to increase the efficiency of several activities, improve the quality of products and services and improve internal and external communication.

Data object
The data objects in the CycloMedia case are 360 degrees panoramic images. These so-called cycloramas are systematically acquired from all public roads with a standard interval of 10 meter. Furthermore, the imagery is geo-referenced and commonly delivered with tools for a seamless integration with the customer’s GIS application. CycloMedia has a continuously growing number of cars with a dedicated camera system mounted on the roof (Figure 24). This system has been developed in-house, including the software for processing the raw image data daily delivered by each car.

Figure 24 – A car with a camera system mounted on the roof.

The images are captured by a digital image sensor and have undergone several alterations. To be precise: several images are stitched together based on calibrations and by using specialized algorithms. The images are additionally both horizontally and vertically adjusted to adapt to the car and camera tilt angles. The final result is a spherical correct image when loaded in a proper viewing application. Figure 25 shows a stitched Cyclorama. In Figure 26 this image is projected on a cylinder.

Figure 25 This is the final result of the image processing. Image are seamlessly stitched and form a single image.
Figure 26 - In this visualization the cyclorama from Figure 25 is projected on a cylinder. Actual Cycloramas made by CycloMedia are even more complex and can be projected on a sphere. The ‘regular’ presentation of a cyclorama is from inside this sphere, showing only a square window in which the user can navigate.

Spatial dimension

The location reference consists of a location on the earth, and a certain height level. Until now the working area was limited to the Netherlands, and thus the local projected coordinate system, the Dutch National Grid (‘Rijksdriehoeksstelsel’ in Dutch) was used [Kadaster]. Since the company has ambitions to expand its working area to other countries, a system with a wider coverage is preferred. A newly developed recording system (DCR7, patents pending) already uses high precision navigation, and is capable of delivering data in a geodetic coordinate system with world-wide coverage: the WGS84 system. Each cyclorama has one single geospatial coordinate: the RecordingLocation. The additional altitude value is the height in meters relative to the (standard geoid of the) earth.

Figure 27 The globe presented here gives an impression of the model of the World Geodetic System (WGS), a coordinate system which covers the entire earth. The marked location is situated somewhere in Turkmenistan.

The spatial reference, the RecordingLocation, corresponds with the exact position of the camera, and is therefore located in the precise ‘center’ of a viewed Cyclorama. In the recording process another important measurement is performed, namely the horizontal direction. This data can be used to rotate a viewed Cyclorama to point of interest. It can also be used to draw an arrow on a map to indicate the direction the user is currently looking. In Figure 28 the RecordingLocation, Altitude and Direction in relation to the information object are presented.
Figure 28 The spatial reference represents the location of the image sensor. This reference, called RecordingLocation, should be located in the center of the cylinder of sphere when the data object is presented to the user, as shown in the above images. In the left image the horizontal direction, at which the data object internally starts is presented by an arrow. This is normally the north direction. In the image on the right the additional altitude level in relation to the data object is shown.

Temporal dimension
All Cycloramas include a valid date and a valid time value as specified by ISO 8601. We will call the temporal reference of the data object the RecordingDateTime. The values are provided by the computer which controls the image capturing in the car. The date part conforms to the Gregorian calendar, which is the most widely used calendar in the world. It consists out of three components: a year, a month and a day value. The time part includes values for hour, minutes and seconds [ISO8601].

Figure 29 Time zones are related to the movement of the earth in relation to the sun. For historical reasons, the reference point is Greenwich, England. The zones are additionally linked to country borders and are therefore quite complex [ICU].

The date and time are furthermore related to the location of the image. This is because of the existence of time zones and daylight savings. Figure 29 gives insight in the complexity of time zones. Figure 30 visualizes the RecordingDateTime in relation to the time dimension. It depends on the intended use of the collection in what way the date and time should be stored. For now, it is good to mention that a collection-wide approach should be used, and that this issue is recognized when utilizing the collection.
Figure 30 The temporal reference, RecordingDateTime, refers to a unique moment on the local or international time line.

Summary
Each data object thus includes a spatial and temporal reference. In Figure 31 the spatial references of a set of cycloramas is projected on a Cartesian grid. By doing this, a map presenting the locations where images have been taken is created. The temporal references are additionally used to add some color variance. For this, each day has been assigned a different color. This presentation gives further insight in the collection. It becomes clear that the images in this area where taken over a broad time range. Besides that it becomes clear that on several locations have been recorded several times.

Figure 31 This image shows the spatial references of a set of Cycloramas projected on a Cartesian grid. Additionally a different color is used for each date.

The visualization in Figure 31 shows us an important aspect of the collection, namely the dynamic content as each day new cycloramas are produced and added to the collection.

---

3 Which is a correct representation since the set contained spatial coordinates in the National Dutch Grid format.
Appendix B - Use Case Model

- Retrieve object → include → Check permission
  - Check maximum number of views and access to the object

- Search for relevant objects
  - Include relevant objects
  - May include various types of spatial and temporal condition

- List accessible objects
  - Include based on chronological preference

- Cluster objects
  - Subtract restricted objects from authorized objects

- Add object
  - Include maximum views in period

- Verify object validity
  - Check for accurate and valid data, time, and geocoded location

- Add newest cadastral regions
  - Objects matching a restricted conditionset can only be accessed by authorized clients

- Analyse coverage

- Add client
  - Authorize a client to a set of conditions

- Add authorization
  - Extends

- Define contract
  - Extends

- Define temporal condition
  - Defined by time, or cyclical constraints

- Define spatial condition
  - Defined by geographical regions

- Use pre-defined cadastral region

- Account Manager
Appendix C - Model criteria

Collection criteria
1. The collection consists of information objects O with a spatial reference OS and a temporal reference OT.
2. The collection contents are dynamic, that is: new, static objects O are added to the collection continuously (or periodically). Once in the collection objects do not change and won’t be removed. The collection thus forms a geoarchive.
3. All objects O are owned by a single party P and provided to subjects S.

Authorization criteria
4. For each subject S individual contract C exists that defines the licensed set of objects L.
5. A contract C, may include a spatial condition SC and/or a temporal condition TC.
6. A spatial condition SC contains a spatial operator SCO and a definition of a spatial geometry CSD.
7. A temporal condition TC contains a temporal operator TCO and the definition of a timestamp CTD.
8. A subject S is granted access to an object O from the collection if both a spatial condition CS and temporal condition CT of a single contract C are satisfied by the spatial reference OS and the temporal reference OT.

Retrieval criteria
9. Subject S can perform retrieval tasks T on their licensed set of objects L.
10. The expression X of a retrieval task T can contain a spatial component XS and/or multiple temporal components XT, and/or a spatial-temporal component XST.
11. The spatial component XS can be categorized as one of the following:
   - nearest-neighbor XSN, in which the nearest object O relative to a spatial geometry point is searched for.
   - within window XSW, in which only objects O within the visible viewing range of a GIS display are searched for.
12. A temporal component XT can be categorized as one of the following:
   - Before timestamp XTB, in which only objects O with a timestamp OT before a provided timestamp are searched for.
   - After timestamp XTA, in which only objects O with a timestamp OT after a provided timestamp are searched for.
13. The spatial-temporal component XST matches a natural phrase for which objects O have to be linked to each other, based on their spatial references OS and temporal references OT. An example is the preference for only retrieving the newest set of objects that cover a certain area.
14. A system implementing the model should be able to complete an average retrieval task within a certain time frame TF. This time frame is based on the use of a dynamic user interface as currently often seen in online mapping applications.
Appendix D - Sample SFS geometries in Oracle

Constructor definition of sdo_geometry:

```sql
CREATE TYPE sdo_geometry AS OBJECT (
    SDO_GTYPE NUMBER,
    SDO_SRID NUMBER,
    SDO_POINT SDO_POINT_TYPE,
    SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
    SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

- The SDO_GTYPE attribute indicates the type of the geometry.
- The SDO_SRID attribute can be used to identify a coordinate system (spatial reference system) to be associated with the geometry.
- The SDO_POINT attribute is used specifically for defining a SDO_POINT_TYPE object type.
- The SDO_ELEM_INFO attribute is defined using a varying length array of numbers. This attribute lets you know how to interpret the ordinates stored in the SDO_ORDINATES attribute.
- The SDO_ORDINATES attribute is defined using a varying length array (1048576) of NUMBER type that stores the coordinate values that make up the boundary of a spatial object.

Below several types of geometries are presented.

**Point Geometry**

```sql
SDO_GEOMETRY(
    2001,
    8307,
    SDO_POINT_TYPE(5.13622867060396, 52.0727168791692, NULL),
    NULL,
    NULL));
```

**Polygon Geometry**

```sql
MDSYS.SDO_GEOMETRY(2003,
    8307, NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1, 3, 1),
    MDSYS.SDO_ORDINATE_ARRAY(5.13622867060396, 52.0727168791692,
    5.14387349255666, 52.0658238683832,
    5.13837357117344, 52.0635923251691,
    5.12639466833892, 52.0650556495838,
    5.12982312791572, 52.0689348103066,
    5.13622867060396, 52.0727168791692)
)
```

**Window-polygon Geometry**

```sql
mdsys.sdo_geometry(2003, 8307, NULL,
    mdsys.sdo_elem_info_array(1, 1003, 1),
    mdsys.sdoordinate_array(
        ?west_border, ?south_border,
        ?east_border, ?north_border
    )
)
```
Appendix E - Spatial-Temporal Query

```sql
SELECT IMAGEID, RECORDINGDATE, V.X LONGITUDE, V.Y LATITUDE,
ROUND(current_date)-ROUND(RECORDINGDATE) AGE
FROM bart.images_authorized,
    TABLE(SDO_UTIL.GETVERTICES(bart.images_authorized.recordinglocation)) V
WHERE SDO_FILTER(LL, mdsys.sdo_geometry(2003,8307,NULL,
    mdsys.sdo_elem_info_array(1,1003,1),
    mdsys.sdoordinate_array(?
west_border,?south_border,
?east_border,?north_border
    ),'querytype=WINDOW') = 'TRUE'
AND IMAGEID NOT IN(
    SELECT U2.IMAGEID ID2
    FROM bart.images_authorized U1, bart.images_aut horized U2
    WHERE SDO_FILTER(LL, mdsys.sdo geometry(2003,8307,NULL,
        mdsys.sdo_elem_info_array(1,1003,1),
        mdsys.sdoordinate_array(?
west_border,?south_border,
?east_border,?north_border
    ),'querytype=WINDOW') = 'TRUE'
    AND SDO_WITHIN_DISTANCE(U1.LL,U2.LL, 'distance="5"')='TRUE'
    AND U1.IMAGEID!=U2.IMAGEID
    AND U1.RECORDINGDATE - U2.RECORDINGDATE > "10"
)
)
```

Additional information

A mock-up of the clustering method is visualized in Figure 32. The small dots present the data objects in the collection. The cylinders present the spatial-temporal cluster area of each object (only a few are drawn). The objects with a red color represent objects that fall inside the cylinder of a newer object (object 3). The green objects form the set of result and are additionally projected on the upper surface of the cube as blue dots. The spatial distance defines the spatial area that an object covers. Not visible is the temporal distance, that prevents objects with a very small temporal difference from being excluded from the result-set.
Figure 32 - Visualization of a complex retrieval task which includes a spatial-temporal analysis.

The complex retrieval task needs additional interface components for the various parameters. An example interface is showed in Figure 33. Using the map the user can define his spatial area of interest, and using the sliderbar below the map the temporal dimension can be restricted. The user can choose between viewing all objects, or a selection. Two types of selections are offered, a covered set of newest objects, or a covered set of oldest objects. The sliders for the temporal and spatial distances can be used to modify the density of the resultset.

Figure 33 - Possible user interface for the proposed retrieval task.
Appendix F - Example spatial-temporal task - Example spatial-temporal task

An authorized area in detail.

All authorized cycloramas in the area are projected on the map.

This is the result of the old ‘dataset approach’ to cover an area with the newest cycloramas: gaps appear on places where no new cycloramas were made during the last recording session.
Utilization of the new architecture: The area is dynamically covered with the newest cyclorama available. Places where no new cyclorama were made are filled with the older ones. It thus creates a set that completely covers the area.

The dynamic approach is realized by clustering available cycloramas based on their mutual spatial and temporal distance. This can be visualized by drawing cylinders.

A detailed view of the clustering method. A cyclorama that falls inside another cyclorama’s cylinder, is ‘expired’, and can be removed. Any cycloramas that don’t fall inside the cylinder of another cyclorama should be in the result-set.