Integrating Crime Data by the Use of Generic Data Models

Dirk Frosch-Wilke
Institute of Business Information Systems
University of Applied Sciences Kiel
Kiel, Germany
e-mail: dirk.frosch-wilke@fh-kiel.de

Lennard Scheffler
Graduate of
University of Applied Sciences Kiel
Kiel, Germany
e-mail: l.scheffler@online.de

Abstract. Effective and efficient crime data analysis can facilitate police work and could increase success rates in crime fighting and crime prevention (e.g. predictive policing). Relevant crime data are usually stored in various heterogeneous systems within different police organizations and other federal agencies. For this reasons many crime fighting agencies are eager to build Data Warehouse Systems for integrating crime data and providing a single data source for crime data analysis and crime data mining. But these Data Warehouse projects are often faced with data quality problems due to incomplete integration of relevant crime data. The reason for this is often the variety, variability, and granularity of crime data in different IT-systems and the inflexibility of the data models used in Data Warehouses to handle these complex data characteristics. In this paper we apply concepts of Generic Data Modeling and Data Model Patterns in order to build data models for crime data which allows complete and consistent integration of crime data in Data Warehouses.

Keywords-Relational Data Modeling; Data Warehouse; Generic Data Modeling; Police Data, Data Model Pattern

I. INTRODUCTION

The research about Business Intelligence and Data Warehousing focused mainly on business [1]. However, the knowledge about how to integrate and analyze data is portable to other contexts. Especially the public sector that stores and manages many different data has now taken advantage of Business Intelligence concepts [2], [3], [4]. This work deals with how to provide high quality data models for the police context. Even though many police departments have noticed the benefit of data analysis, they often suffer from insufficient data integration and analysis options. Though the reasons for this are sometimes project specific, a more general reason is the quality of the data models especially in the crime fighting sector. Therefore this paper particularly deals with how to develop flexible and solid data models in order to integrate crime data from various sources on the long run.

The paper is organized as follows. In Section II we shortly describe the Application Domain. Section III provides the theoretical background for data model patterns as well as for Generic Data Modelling and thereby describing the Knowledge Base of our research.

In Section IV we present some of our design artifacts for high quality crime data models as well as the evaluation of these artifacts. The paper ends with a conclusion and prospects regarding further research activities.

II. APPLICATION DOMAIN

Even though this research results in abstracted data model solutions that can be applied to various projects and contexts regarding to the police domain, the starting point has been a data warehouse project of some federal police department in Germany. Since the year 2012 both, the Business Intelligence Competence Center (BICC) as well as the Business Intelligence department of the federal IT-service provider maintain a data warehouse system in order to support strategic and operational crime fighting activities. This system is based on Inmon's enterprise architecture [1] running on an Oracle database system. It integrates various operational data sources into a relational core area and provides different multi-dimensional data marts for analysis purposes. Outcomes can be visualized ether in charts, crosstabs or geographical maps. This system especially in terms of standard reporting is used by police officers as well as management and controlling.

The challenge of this project is to build flexible and consistent data models, which allow to easily integrating existing data sets as well as new kinds of data. The crime-fighting sector stands out due to a huge amount of different data. This means, documenting a criminal case brings up varying entities and objects depending on the particular circumstances. Thus, examining a single case, the amount of data is not down to a large number of similar objects but too many objects of different classes, e.g. persons, things, locations, actions and so on. For this reason, detailed relationships between objects usually can add most of the value for data analysis. Only inquiring objects of one class without considering relationships to other entities will not lead to satisfying results. Summarized, a data model for integrating crime data has to care about both, providing well-structured objects as well as detailed relationships that can handle most of possible object correlations.

Although this is a challenging task on its own, the developer has to care also about maintaining the data model in a flexible way in order to fit future requirements and simultaneously lower the complexity towards the end-user. Because of different reasons the mentioned project sticks to multi-dimensional modeling of data marts, where it can be difficult to transform the complex relational core model into a valid and high-performance multi-dimensional model. In particular, the many n:m-relationships can affect the size and complexity of fact tables enormously. Lastly, it is not possible to develop data models (neither relational nor multi-dimension) without profound knowledge of the problem domain.

III. KNOWLEDGE BASE

A. Data Model Patterns

Back in the beginnings of software engineering in the sixties of the last century software development was understood as an artistic task [5]. Friedrich L. Bauer was the first to introduce the idea of a software developer as an engineer in 1969 [6]. Since then standardization in methods, processes, roles and techniques increased by following the example of conventional engineering. One of the most influential advantages has been the development of reusable parts, so called *patterns*, basically invented by Christopher Alexander [7]. While the use of patterns has already been known for many years in programming [8] it is now assigned to the area of conceptual data modeling e.g. by David C. Hay [9] or Len Silverston [10] and has been applied e.g. for dimensional modeling to the Data Warehouse context [11].

In the context of data modeling a pattern is understood as a template or guidance for developing parts of a data model in order to store data of a certain domain. Every pattern is abstracted to be reusable for a class of similar problems. Moreover, a pattern consists of three parts. The *context* defines in which condition the pattern can be used. The *problem* describes the challenge that has to be solved. Lastly there is an *abstract solution*, which has to be concretized by the user [12]. The probably greatest advantage in using data model patterns is to solve a problem once and then apply this solution to multiple data models. This leads to a better understanding of data models and decreases development costs. Furthermore, high quality patterns can highly increase the strength and flexibility of data models.

B. Generic Data Modeling

The benefits of patterns can increase data model quality, if the pattern itself is of high quality. To create significant data model patterns, there is a need for methodic definition. In 2007, Stephan Schneider introduced the concept of *Generic Data Modelling* as methodological background [13].

Many developers presuppose an adequate grade of their data models, which results in insufficient quality management. This approach often tempts to cover weaknesses of data models, which ends in low quality results. On top of that, these vulnerabilities are frequently not noticed before trying to embed new requirements. Therefore, the method of generic data modeling integrates the quality management into the process of modeling. Basically, a developer has to follow eleven rules exceeding known strategies like normalization (c.f. table 1). Following these rules is one important activity in achieving high quality data models. But it is also necessary to separate the concepts of *nature* and *roles*, which many data modeling professionals are lacking [13].

TABLE 1 RULES OF GENERIC DATA MODELING [13, p. 395 ff.]

Rule	**Description**
completeness	The data model represents the whole problem domain or the necessary parts.
correctness	All data model items are consistent and do not need to be interpreted.
minimality	The model is free of redundancy; no part could be removed without losing information content.
understandability	The model is significant to the reader by using meaningful naming for entities, properties and relationships.
simplicity	The data model is not unnecessarily complex.
flexibility	The data model enables easy integration of changed requirements or conditions.
stability	The data model offers ways to implement changed requirements or conditions without changing structure.
modularity	The data model is based on consistent model parts.
reusability	Model parts can be adopted directly or slightly modified to other model parts.
ability for integration	Data model parts can be combined to a complete model.
ability for implementation	The data model can be implemented on a database system.

A simple example may explain both concepts: In the problem domain of crime fighting the entities suspect, witness, and victim are obvious. A quick and even common data modeling solution would be to model these three as separate entities. But of course these identified entities have many common attributes like name, address, date of birth and so on. For this reason the abstraction of parts of each of these entities to a generalized entity named *person* is possible. *Person* reflects the *nature* of the former mentioned entities. Understanding these entities as *roles* of a *person* would complete the data model and make it even closer to reality (cf. Fig. 1). Moreover, separating between nature and role can help avoiding *hidden redundancies*. This becomes clear while looking at a suspect for instance, who is witness in the same record. By using the role concept, it is possible to store the key attributes of the person only once and then relate it to its roles. The other way would expect to store the person for each of its roles, which will expire in data inconsistencies.

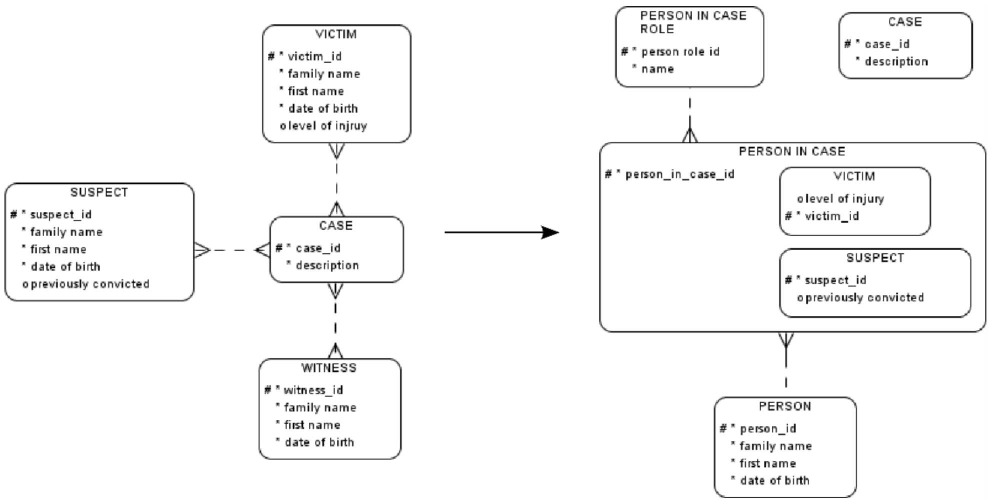

Figure 1. Abstraction by example

IV. Design Artifacts

The following part of this article explains some of our developed patterns for crime data integration within the above mentioned project (c.f. Section II). The presented data models are not necessarily complete but they build the fundamental architecture which allows an easy extension of the models by following the explained rules.

A. Characteristics

One of the main challenges in data modelling for crime fighting is the complexity of police work. The key to effective crime fighting is to store as many facts as possible that might be important for solving crime cases. In consequence, crime data is characterized by a very high level of granularity and heterogeneity in order to fulfill this requirement.

The initial data modeling process cannot account for every possible fact that might be relevant later on. Therefore, the data model has to fit abstraction requirements in order to store every relevant data in future. Otherwise, the limitation of storable attributes for describing a suspect for example could end up in bad analytical and crime fighting results because of information loss.

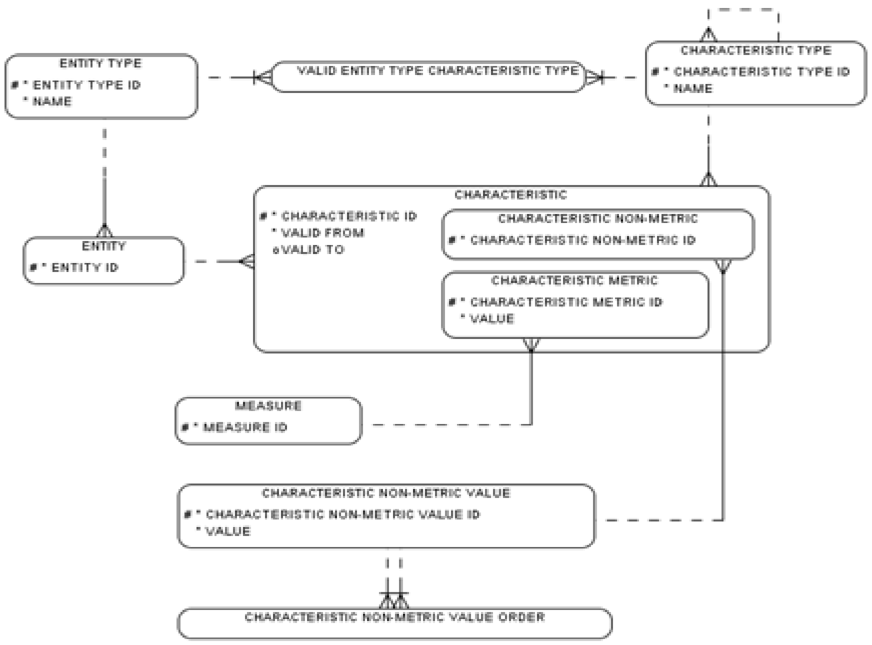
ENTITY TYPE
# * ENTITY TYPE ID
* NAME
VALID ENTITY TYPE CHARACTERISTIC TYPE
CHARACTERISTIC TYPE
# * CHARACTERISTIC TYPE ID
* NAME
ENTITY
# * ENTITY ID
CHARACTERISTIC
# * CHARACTERISTIC ID
* VALID FROM
o VALID TO
CHARACTERISTIC NON-METRIC
# * CHARACTERISTIC NON-METRIC ID
CHARACTERISTIC METRIC
# * CHARACTERISTIC METRIC ID
* VALUE
MEASURE
# * MEASURE ID
CHARACTERISTIC NON-METRIC VALUE
# * CHARACTERISTIC NON-METRIC VALUE ID
* VALUE
CHARACTERISTIC NON-METRIC VALUE ORDER

Figure 2. Highly abstracted characteristic pattern

.

Further on, different operational police applications will store different facts depending on their purposes and orientation. To be able to integrate all these operational crime data as well as to be flexible for integrating new criminological relevant characteristics we have developed the data model in Fig. 2.

The data model pattern in Fig. 2 is all about managing characteristics and can be applied to many different entities. It is based on the idea that a characteristic consists of attributes, e.g. the *color* of a car, and its specification, e.g. *red* [14].

In the pattern's center there is the CHARACTERISTIC representing a property. The CHARACTERISTIC is separated into METRIC and NON-METRIC depending on the kind of possible specifications. METRIC CHARACTERISTIC is used to manage properties with countable specification. In order to enable interpretations of distances, a METRIC CHARACTERISTIC is optionally related to a MEASURE. In contrast, NON-METRIC CHARACTERISTIC can be used to store uncountable measures. To avoid hidden redundancies, specifications are stored within a separate entity in this case.

The attributes VALID_FROM and VALID_TO take care of the time aspect of properties. Some properties may only be valid for a period of time, especially if they are roles. Other properties may be constant in property but changeable in specification, which certain timestamps can handle as well.

By using a power type like CHARACTERISTIC TYPE it is possible to add categories or hierarchies to

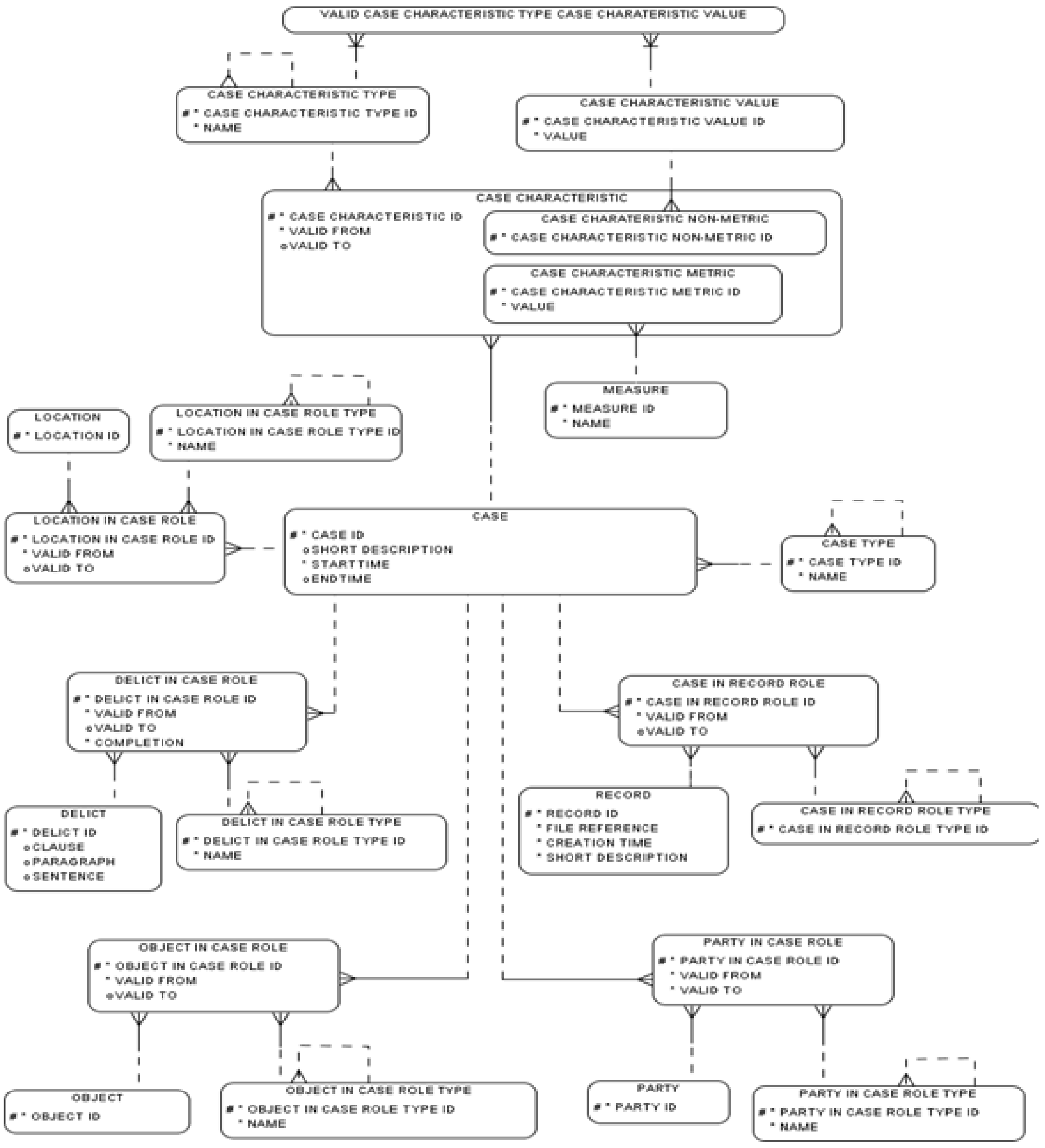

Figure 3. Cases and their contextual relationships

store characteristics, which is very useful especially for drilling-functions in data analysis.

By implementing this pattern, a data model can store affected entities in a much more flexible way. For example, decorating parties (this means persons, organizations, etc.) with characteristics enables you to store every single important fact about a suspect. This could be the body height, hair and eye color as well as clothing or behaviors. Every new characteristic can be easily applied by just adding a new characteristic type on data level without changing the models structure.

B. Records and Cases

Our data model about records and cases in Fig. 3 is less abstract in terms of different fields of application.

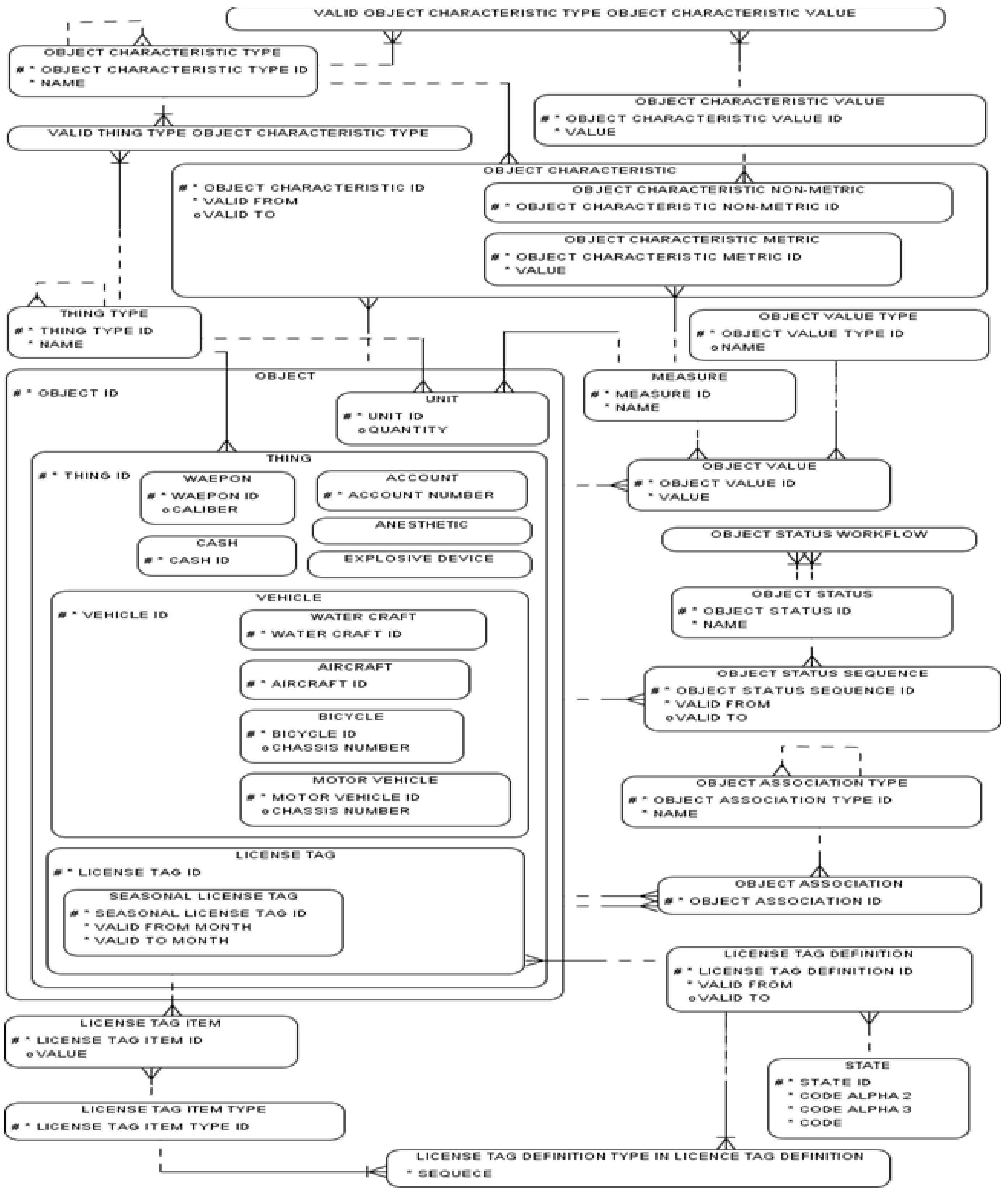

Figure 4. Things pattern

Nevertheless, it is built out of abstract data model patterns so that it is very flexible within the context of crime data.

A RECORD is the central element of operational police work. It is the representation of an old-fashioned file or document that bundles every relevant items or cases. It has no direct crime fighting significance but it is used to structure data on a higher level, especially in operational work.

More important for crime data analysis is the CASE entity instead. The CASE can be understood as every committed criminal action. Because of that, a CASE contains contextual relationships to nearly every other entity stored in the whole data model. For this reason, it has to be possible to store LOCATIONS as well as PARTIES or THINGS depending on their role.

This kind of modelling is what achieves the needed flexibility. By adding roles or entity types one can easily expand the model without structural changes.

We used the characteristic pattern (Fig. 2) within the case data model (Fig. 3) in order to store the way of committing crimes or other properties that might be unknown now.

Different from the characteristic model, this pattern mainly adds value by holding relationships instead of objects details. As we described in Section II, this is one of the key tasks for crime fighting data models. This data model allows storing a very detailed view of cases and their circumstances. Looking at a case one can extract relevant information like e.g. who was concerned, who is suspected of having committed an offence, or which things were used. This means the CASE entity enables allowing getting a full overview of a committed crime. From a technical perspective, this gets possible by using *contextual roles* which allows setting up one single relationship on a structural level that can be specified on the object level by using role types. Now we can store new kinds of relationships easily by just adding roles that concretize an existing relationship.

C. Things

Among the administrative level shown in Section IV B one of the main tasks in police work is managing things corresponding to a case. This may contain stolen or damaged goods, evidence or anything else. The main difference between the police and conventional business context is that criminological data processing has to be able to store nearly every kind of thing at a very high level of granularity. Thus, a data model managing things has to be very close to the real world. Remembering that one of the key steps in data model development is to define the relevant real world cutout, we now have to enable the database to store nearly any real world requirement.

Fig. 4 is an approach of meeting all the mentioned goals. In the center of this figure the key entity THING representing any possible physical item. To give a glue of what a thing can be, the entity exemplarily contains some derivatives like WEAPON, VEHICLE or ANESTHETIC. Of course, every sub-entity can be specialized further on if needed.

As the police not only deal with single items, the model has to enable the database to store UNITS. This is to be explained by means of an example: Now we can store one stolen notebook by using the entity THING as well as a whole bunch of stolen notebooks by using the entity UNIT. Because both entities are very similar in shaping and share many relationships to other entities, they are both generalized to OBJECT, which mainly is an expedient to hold relationships centrally.

In conventional modeling every sub-entity of THING would have to hold its nature properties. Still, this would be an acceptable and solid approach. Because of the variety, it would be hard to predefine these properties as well as to decide on the optimal abstraction level. This is why we decided to outsource properties of a thing to the already introduced pattern of characteristics.

The THING model is a very good example of how to make use of universal data model patterns like characteristics and the strength of generic data models. The model stays with conventional modeling and database techniques but is also very near to real world requirements. This also clarifies that multi-structured data is not necessarily needed in order to add flexibility to data models.

The THINGS pattern combines detailed object descriptions with many relationships to other entities and the fields of application are very wide spread.

V. Conclusion

The concepts of patterns have already been known for many years in software engineering. By adopting these ideas and transferring them to the process of data modeling especially regarding to the police domain, we can increase the quality of data models while simultaneously decreasing development costs. There is still a need of methodological background. The method of Generic Data Modeling enables the developer to integrate quality management into the process of modeling.

In this paper we illustrated the use of Generic Data Modeling in the context of data modeling for crime data. Generic Data Modelling allows us to develop flexible data models that scale with the agile requirements for crime fighting and offer the needed strength to be a solid platform for data analysis at the same time. Referring to the use of such a data model for police data warehouse, it becomes possible to integrate data from different heterogeneous sources easily. Because of the abstraction that is based on a target-oriented analysis of the problem domain instead of technical requirements this model can perfectly be used as a structural protection layer for different data marts.

Further research work needs to deal with demonstrating the performance of data analysis based on such abstract data models, quantifying the benefits of using patterns for data modeling and transferring Generic Data Modeling concepts to other application domains as well.

REFERENCES

[1] Inmon, W. H.: Building the data warehouse, 4th edition, Indianapolis (2005)

[2] Hartley, K., Seymour, L.F.: Towards a framework for the adoption of business intelligence in public sector organisations: the case of South Africa, in: Proceedings SA-ICSIT’11, pp. 116-122 (2011)

[3] Kim, G.-H., Trimi, S., Chung, J.-H.: Big data applications in the government sector, in: Communications of the ACM, Vol. 57, No. 3, pp. 78-85 (2014)

[4] Chen, H. et al.: Crime Data Mining: A General Framework and Some Examples, in: Computer, Vol. 37, No. 4, pp. 50-56 (2004)

[5] Ludewig, J., Lichter H.: Software Engineering, Dpunkt, Heidelberg (2010)

[6] Bauer, F. L.: Software Engineering. Wie alles begann, in: Informatik Spektrum, Nr. 5, pp. 259-260 (1993)

[7] Alexander, C.: A Pattern Language: Towns, Buildings, Constructions. Oxford University Press (1978)

[8] Gamma, E: Design patterns. Elements of reusable object oriented software. (1995)

[9] Hay, D. C.: Data model patterns. Conventions of thought, New York (1996)

[10] Silverston, L., Agnew, P.: The data model resource book, Indianapolis (2009)

[11] Schneider, S., Frosch-Wilke, D.: Analysis Pattern in Dimensional Data Modelling, in: Kannan, R., Andres, F. (ed.): Data Engineering and Management – Second International Conference, ICDEM 2010; LNCS Vol. 6411, Springer, pp. 109-116 (2012)

[12] Tešanovic, A.: What is a pattern?, URL: http://st.inf.tu-dresden.de/Lehre/dpf/IntroductoryPapers/tesanovic-WhatIsAPattern.pdf, April 24, 2014

[13] Schneider, S.: Konstruktion generischer Datenmodelle auf fachkonzeptioneller Ebene im betrieblichen Anwendungskontext. Methode und Studie. Dissertation. European Business School (2007)

[14] Summerford, J.: Neither Universals nor Nominalism. Kinds and the Problem of Universals, in Metaphyica, No. 5, pp. 101 – 126 (2003)