A Model-Driven Development Framework for Geographical and Relational Databases Systems

Carlos Eduardo Pantoja
CEFET/RJ
Universidade Federal Fluminense
pantoja@cefet-rj.br

Tielle da Silva Alexandre and José Viterbo
Universidade Federal Fluminense
tiellesa@id.uff.br, viterbo@ic.uff.br

João Victor Guinelli
Petrobrás
CEFET/RJ
joao.silva@cefet-rj.br

***Abstract*—The database project is the process of engineering a database model from conceptual modeling to database implementation. Existing tools allow the conceptual modeling of relational and geographical databases separately, but none integrates both in a single solution. This paper presents an Model-Driven Development framework for creating relational and geographical database models. The framework comprises an extended relational metamodel to adhere to geographical database concepts present in the OMT-G, an Entity-Relationship modeling tool, Query View Transformation rules between OMT-G and the extended metamodel, and Model-To-Text transformations to generate ANSI SQL/SFS code.**

***Index Terms*—MDA, database, software engineering**

I. Introduction

The database project consists of different abstraction levels that converge to the database implementation using a Database Management System (DBSM) and a structured language. A database system project can be considered a set of sequenced transformations from high abstraction models to a specific platform model. At the early level, the database requirements gather textual business information necessary to create a database structure. Afterward, the database designers, considering a modeling technique, transform this textual business description into a graphical model. The conceptual modeling is responsible for describing the business information model using a graphical notation at a higher abstraction level. It allows an early visualization of the structure of a database and how these structures are related to each other [1].

The next level results from a transformation from conceptual database modeling to a logical model. The logical model considers additional constructions to the conceptual database modeling to avoid anomalies, redundancy, and inconsistency. This level does not consider any implementation characteristic using a specific technology or language. However, it can consider the kind of database modeled, e.g., relational databases. The physical model is the consequence of transforming from the logical model considering a target DBSM and a database language. This level considers some specific technologies for the implementation of the modeled database.

There are several modeling languages and notations for modeling a relational database, such as ER [2]; Crow's Foot [3]; and IDE1FX [4]. Several modeling options exist for a geographical database system, such as UML-GeoFrame [5], [6] and the OMT-G [7]. However, these languages do not offer a solution integrating different models and approaches. In general, a geographical database system cannot be projected using a relational notation; otherwise, it is possible to model a relational system using some constructions present in geographical notation since it extends the relational concepts.

Conversely, Model-Driven Engineering (MDE) provides a software development process based on model transformations that separate the business modeling from specific technologies until effective software implementation. The Model-Driven Architecture (MDA) specifies abstraction levels from technologically independent models to a specific platform model to assist software development. The MDA uses a Computational Independent Model (CIM); a Platform Independent Model (PIM); and a Platform Specific Model (PSM) [8].

Applying the MDA in the software development process implies using metamodels, which define the concepts of another model at a lower level. Several metamodels intend to provide concepts for a database system, such as the Common Warehouse Metamodel (CWM) [9] for interchanging information between the Meta Object Facility (MOF) [10], Unified Modeling Language (UML) [11] and XML Metadata Interchange (XMI). It provides a generic model for several languages and notations, including the relational model. However, it does not include any geographical concept. It provides a simple mapping between relational and oriented-object concepts, the Enhanced Entity-Relationship (EER) [12] metamodel, which provides concepts for all ER constructions using Chen's notation. Although a case tool supports the metamodel, it does not provide constructions for other modeling languages, such as Crow's Foot; and the Generic Database Modeling Metamodel (GEDBM) [13], which gathers concepts of relational database modeling languages and several notations. The metamodel is supported by a semi-automatic tool and a SQL code generator but is strictly directed toward relational concepts.

This paper presents a framework for designing database systems, which uses a metamodel for relational and geographical database system projects that gathers the concepts of several modeling languages and notations. The relational part was constructed by observing the extant modeling languages and notations used in the database development process. The geographical part of the metamodel was constructed by ob-

10.18293/SEKE2023-018

serving the concepts of the OMT-G model. The methodology is composed of the revised GEDBM, which can be instantiated by a set of Query-View-Transformation (QVT) [14] rules from an adherent modeling language and; an existing CASE tool or by a new one constructed from the GEDBM concepts. A set of QVT rules from the OMT-G model to the GEDBM will be defined to prove the validity of geographical databases.

After that, a set of Model-To-Text (M2T) [15] rules for ANSI/SQL and SFS/SQL specifications can be generated from the GEDBM independently of the modeling language used; the GEDBM will be constructed using the Ecore from Eclipse Modeling Framework (EMF) [16]; the QVT rules will be implemented using EMF's Model-To-Model (M2M); the M2T rules will be implemented using Acceleo [17]; and the graphical ER tool will be constructed using the Graphical Modeling Framework (GMF). We aim to provide a framework for both database teaching and development. Besides, it can be extensible for several notations and modeling languages, amplifying the framework's applicability in various domains. We also provide some set of transformations between different models and target platforms to facilitate the database development process (for both relational and geographical databases).

II. Metamodels for Designing Database Systems

The GEDBM metamodel is based on the conceptual and logical database project and is generic for the most used relational modeling languages and notations. The modeling languages adherent to GEDBM is ER Chen's Notation, Crow's Foot, and IDE1FX. The metamodel consists of meta-classes such as Database, Relationship, Entity, and Field that represent aspects of conceptual modeling languages. Besides, it defines others like PrimaryKey, ForeignKey, and Check that represent aspects of the logical project, such as the uniqueness of a record, referential integrity, and attributes integrity. Thus, it is possible to build graphical tools for relational modeling languages using this metamodel. In the same way, if any tool exists with a modeling language adherent to GEDBM, it can be integrated into the GEDBM core by mapping its concepts.

The OMT-G metamodel is used to model geographical databases. The meta-class Schema contains the meta-class Element and the meta-class BaseRelationship that represent entities and relationships, respectively. The Element is specialized in Conventional, which can be geoObject and geoField. Thus, the created entities can represent three data types that exist in the OMT-G model: non-spatial (Conventional), discrete (geoObject), and continuous (geoField).

Furthermore, in the OMT-G, there are five different meta-class types to represent geographic fields (Network Class, Isoline, Sampling, Tesselation, and AdjacentPolygons), so the class geoField is specialized in five meta-classes that represent these types. The same thing occurs with the class geoObject, this meta-class is specialized in geoObjectWithGeometry and geoObjectWithGeometryAndTopology, and these two meta-classes are specialized too: the first one in Polygon, Point, and Line; and the second one in Node, UnidirectionalLine, and BidirectionalLine. Similarly, the meta-class baseRelationship, which represents the relationship between two entities, is specialized in association, Aggregation, generalization, SpatialAgregation, and cartographicGeneralization.

III. Methodology

In this section, we present the MDD methodology (Figure 1) for relational or geographical database projects. In the relational part we use the GEDBM metamodel, which gathers generic concepts for several modeling languages. Our extended metamodel uses the OMT-G to ensure the representation of geographical concepts. Furthermore, we present a set of M2M transformation rules for mapping concepts from the OMT-G to the GEDBM metamodel and a set of M2T transformation rules, which transforms the instance of the GEDBM metamodel in SQL and SFS code. The proposed methodology relies on the MDA to specify metamodels and transformation sets.

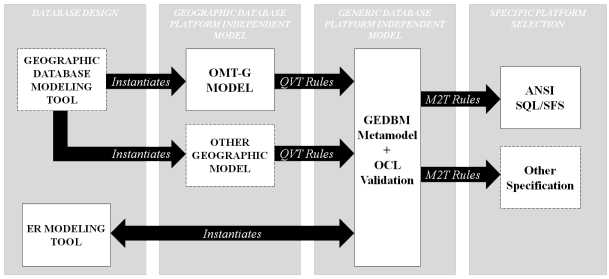

Fig. 1. The proposed MDD approach.

The MDA is a software development approach created by OMG where the metamodel has an important role in software development. In the MDA, the transformation process among models in different levels of abstraction is responsible for the software construction. The MDA approach contributes to the development of applications more independent of technology once models can be combined and new models from new technologies can be later added. It increases the interoperability of the project and facilitates its maintenance.

Our methodology starts by selecting one of the several modeling languages that can be used in GEDBM or the OMT-G model. In this step, it is possible to interconnect existing tools to instantiate these models once their concepts adhere to the ones present in GEDBM. Suppose it is chosen the ER modeling tool. In that case, the GEDBM is directly instantiated (in this case, the modeling tool should be constructed using the GEDBM, or its metamodel should adhere to the GEDBM). If the OMT-G model is chosen, it is possible to execute a transformation using the QVT language between the chosen model and the GEDBM metamodel. Otherwise, the GEDBM can be manually instantiated. The Object Constraint Language (OCL) is used in the metamodel validation to guarantee correctness (with minimum concepts required for GEDBM).

After that, it is possible to generate ANSI SQL 92/99/03 or SFS/SQL standard code using a set of transformation rules that uses the M2T specification. The SFS is a specification proposed by OGC that defines how spatial or vector components of geographical data should be stored in a database; and

how to store, read, query, and update these components using SQL. The PostgreSQL and Oracle Database are widely used DBMS with geographical extensions based on the SFS/SQL.

Besides, other tools, models, and target platforms can be added to the methodology. When it is desired to integrate an existing tool, it must use the GEDBM or the OMT-G as metamodel; or its metamodel must comply with the GEDBM. In this case, it is possible to specify a new QVT transformation from the existing model to the GEDBM. Finally, another target platform can be added by creating a new set of transformation rules in M2T from GEDBM to the target platform.

A. The Revised GEDBM Metamodel

The methodology uses the GEDBM as the core of the MDA approach for database design. However, the original GEDBM is not prepared to support concepts from geographical modeling languages and models. Therefore, we present a modified version of the GEDBM metamodel, where some modifications were performed to add geographical concepts to the GEDBM. The modifications include analyzing if geographical constructions from the OMT-G model adhere to the GEDBM. The metamodel also accepts the min-max notation. The GEDBM metamodel can be seen in Figure 2.

In the OMT-G, entities have different types. So, it was necessary to extend the representation of entities and relationships so that they could represent geographical types that do not exist in relational modeling. For this, it was added the enumerations EntityType and RelationshipType. So, as primary keys and foreign keys are not represented in some modeling languages in the conceptual model, we decided to substitute them. The meta-class Field was modified to a common field (CommonField) or an identifier field (IdentifierField), and they can refer to an Entity. Now, the rules to generate SQL code are simpler: the foreign keys are automatically generated, and the primary keys result from instances of the IdentifierField.

Additionally, the concepts of the weak and associative entity were added to the metamodel. Then, the meta-class Entity was modified to be the parent of common entities (CommonEntity) and associative entities (AssociativeEntity). Furthermore, a common entity can now be represented as a StrongEntity or WeakEntity. So, the metamodel is now formed by a base meta-class DataBase that contains Entities and Relationships with two or more Cardinalities. There is an associated cardinality for each entity, so it is possible to represent the relationships between an entity and itself and between two or more entities. Moreover, an AssociativeEntity refers to exactly two other entities without a relationship between them (the associative entity is a result of the relationship of these two others entities). Besides, an Entity has many Fields that are specialized in CommonFields or IdentifierFields.

B. From OMT-G Model To GEDBM Model

The first set of transformation rules — QVT — uses the OMT-G as the source Platform Independent Model and the GEDBM as the target Platform Independent Model. This transformation set will generate an equivalent GEDBM model with the occurrences of geographical concepts presented in the OMT-G modeling project. The QVT provides the transformation rules because it is a standardized specification for transformations between different models.

Firstly, the root element Schema is selected to be mapped to a Database element in GEDBM using the Schema2DataBase transformation. After, all classes present in a schema are mapped to entities in the target model based on their type. All these classes' attributes are also mapped using the Attribute2CommomField transformation. Identifier attributes are mapped using the Attribute2IdentifiesField transformation. Finally, the geographical relationships are mapped based on their type, with one transformation for each type in the OMT-G.

C. From GEDBM Metamodel To SFS/SQL Code

The second set of transformation rules is responsible for generating the SFS/SQL code from the GEDBM (which was previously instantiated from a relational or a geographical modeling project). The GEDBM is the source Platform Independent Model from this transformation, while the SFS/SQL is the target Platform Specific Model. We use M2T specification because it generates text artifacts from models. The process begins with the ToSQL transformation, which creates the text file of the generated SQL. Next, the ToDataBase transformation generates the name of the database to be created, and the ToEntity maps all entities to tables in the following order: the entities that do not have foreign keys and the entities that have foreign keys of the generated entities.

The transformation PrintRelationshipAsEntity generates the code for the associative entity. The PrintSelfRelationship does the same for self-relationships. ThePrintConventionalEntity generates SQL code, while PrintGeographicEntity, PrintArcNodeTopology, and PrintGeospatialFeature transformations print SFS code. The last transformations generate codes for fields, primary and foreign keys, text and numeric limits, default value, integrity, and check constructions.

D. Discussion

The meta-modeling allows a system construction by defining different abstraction models, starting from a high abstraction model to a particular model. The database project also uses different designing levels, from requirements engineering to a physical project specification using a specific technology. Therefore, using the MDD methodology, it is possible to generate the physical project performing model-to-model transformations from a conceptual model.

The methodology uses a generic metamodel, which allows the use of several relational modeling languages and notations. Once it gathers common concepts from these notations and languages, it can generate graphical representations using a single metamodel. Besides, suppose a graphical representation instantiates (directly or indirectly) the metamodel and exists another graphical representation directly constructed over GEDBM. In that case, it is possible to generate an equivalent graphical representation between these modeling (because the metamodel instance is unique for both modeling in this case).

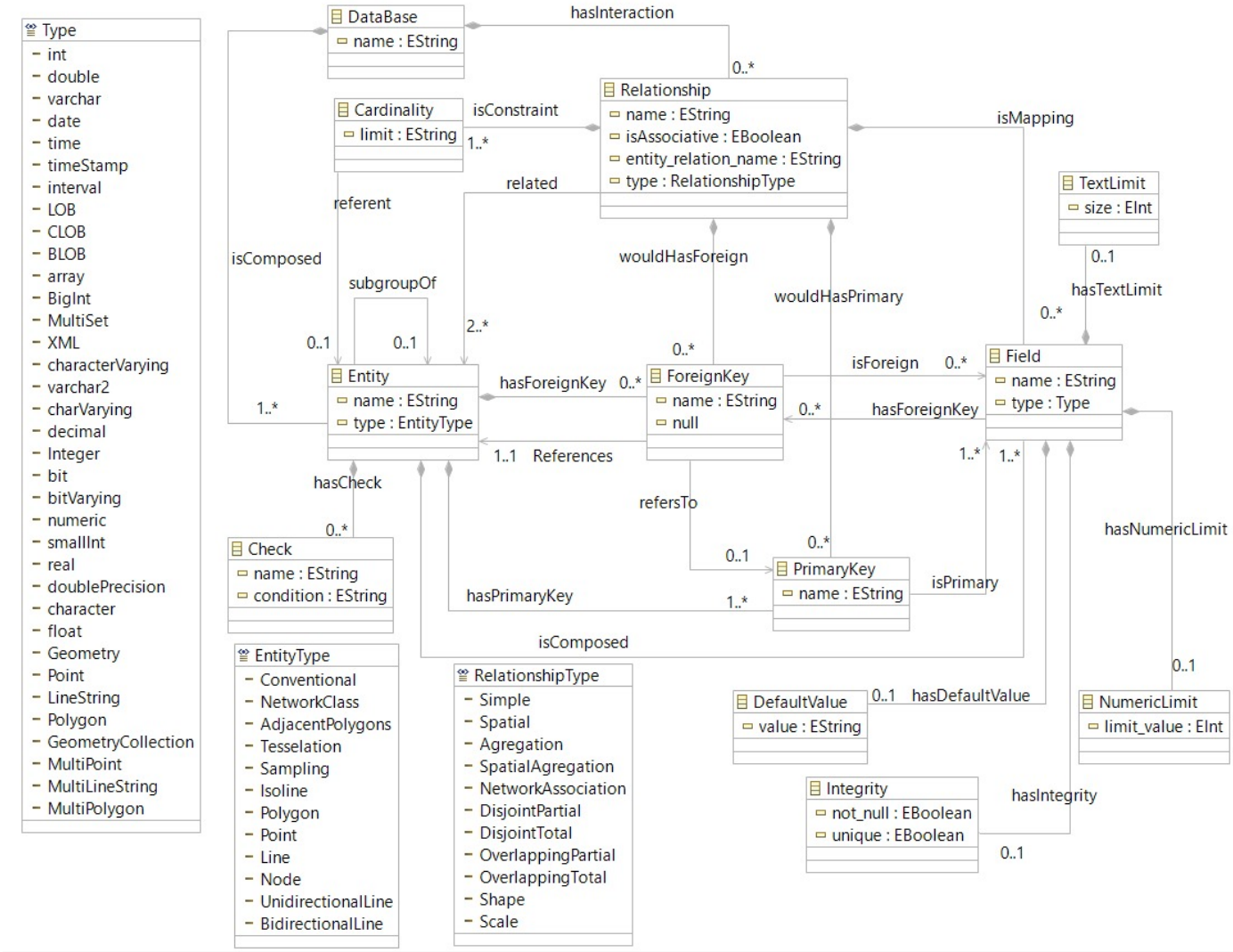

Fig. 2. The GEDBM metamodel.

This feature could be interesting in a database team where the designers do not know or use the same modeling language.

The methodology can also be adapted to other geographical models (e.g., UML-GeoFrame), requiring a new set of transformation rules between the new model and the GEDBM. Then, any existing transformations can be realized from the GEDBM without needing new or modified transformation rules. Likewise, the methodology allows code generation to any other specification (since their concepts are present in GEDBM). In this case, generating new transformation rules for the new specification code is necessary.

IV. Designing a Database Using the GEDBM

This section presents the developed components of the proposed framework for database modeling. We used GMF to develop the graphical environment; EMF to define and create the metamodels; the M2M and Acceleo to generate text artifacts based on ANSI SQL patterns; and OCL for constraining the models. The environment is available for the Eclipse Framework and is a set of plugins. Figure 3 illustrates the environment for developing diagrams using the Entity-Relationship diagram and the palette of items.

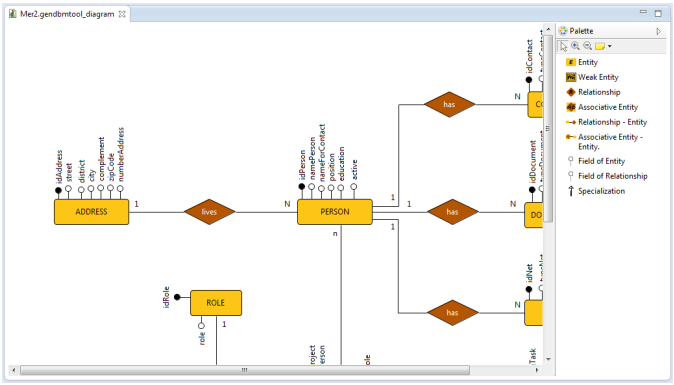

Fig. 3. The graphical tool for ER modeling of the framework.

Once a diagram is modeled, it can access its metamodel, where all the structures designed can be found. The EMF tree diagram can add additional information not graphically represented in the model. For example, fields such as Integrity and text limits can be inserted for each attribute. The instantiated metamodel for a Project Management System example can be seen in Figure 4. It is possible to generate the diagram directly from a modeling instance using the metamodel if the user prefers to instantiate the model in the tree view.

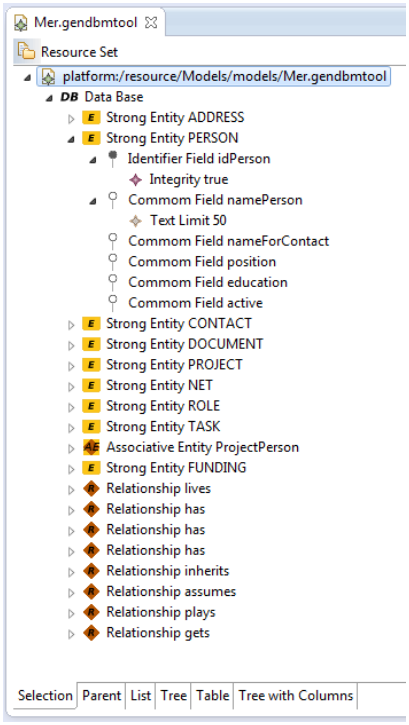

Fig. 4. The proposed MDD approach.

The Figure 5 shows a modeling example of the behavior of the metamodel while using the GEDBM tool to create a self-relationship, a ternary relationship, and simple modeling. It is possible to see (the dotted lines) that every rectangle is instanced as an Entity's object while every diamond is instanced as a Relationship. Cardinality is associated to only an entity, but every relationship has at least two cardinalities constraining them. So, it is possible to generate the cardinality for ternary relations and self-relationships, for example.

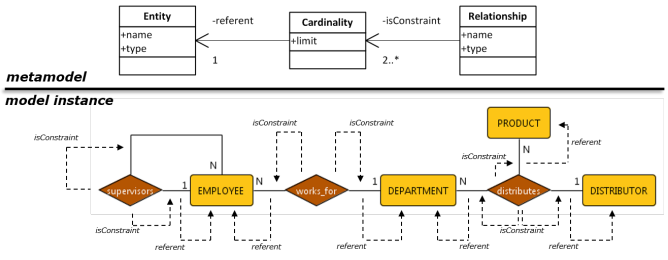

Fig. 5. The metamodel behavior for relationships and cardinality.

Similarly, the specialization concept and the associative entity modeling behavior can be seen in Figure 6. In order to represent the specialization concept, an entity can be a subgroup of other entities allowing the design of a hierarchical structure. The associative entity is referent to exactly two common entities because it is the consequence of a many-to-many relationship. Besides, some concepts are directly instantiated, such as fields and constraints (text limit, numeric limit, default value, integrity, and check).

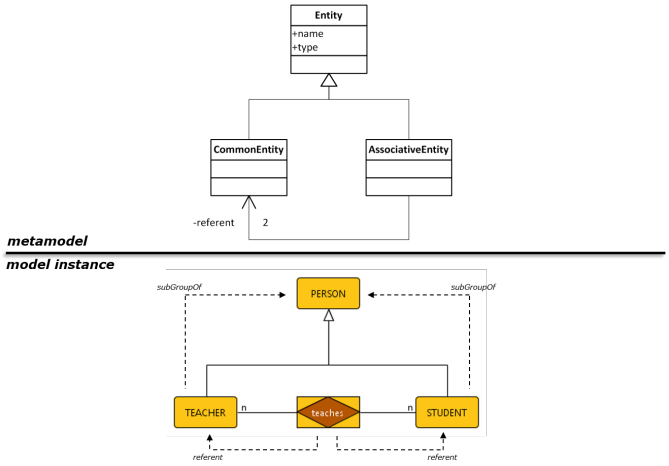

Fig. 6. The metamodel behavior for specialization and associative entities.

The OMT-G diagram is modeled using the OMT-G Designer tool [18], a set of plugins for the Eclipse. This tool is independent of our solution and is used to provide geographical modeling in the proposed framework. The tool was developed using an MDD approach with EMF and GMF. The OMT-G Design tool is responsible for instantiating the concepts of the OMT-G metamodel present in the solution. Once this metamodel is instantiated, it is possible to execute the transformation to the GEDBM using a QVT transformation. This process generates a GEDBM model instantiated with the concepts modeled in geographical notation, and an equivalent relational model can be generated. Then, the SQL/SFS code is generated from the GEDBM instance using Acceleo.

The proposed MDD approach avoids human mistakes since it is automatic and assisted by a CASE tool, demanding only more time from the designer in database system early requirements. When designing a system not using an automated approach, some mistakes can be non-intentionally added to the development process. These mistakes are avoided because the transformations from models to models use formal specifications (QVT and M2T) based on OCL. So, if a concept is modeled in early phases, the same concept will be guaranteed to be present in the final database system.

V. Related Work

The CWM uses the MOF to allow the modeling of object-oriented notations such as UML and the XMI. The relational package of CWM covers the SQL standard and relational DBMS concepts. There is no official tool associated with

CWM, but it has several vendors that implement its core. The revised GEDBM allows several relational modeling languages and has an MDA tool for Chen's notation. It also integrates with the OMT-G geographical model for modeling geographical concepts. The revised GEDBM does not aim to replace or exceed the CWM. Instead, it could be integrated into the CWM since its core is generic for relational languages. Besides, the GEDBM can be easily mapped to the CWM since it adheres to the MOF, extending the scope for geographical databases.

The EER metamodel aims to provide all concepts for the Enhanced Entity-Relationship model [1], which extends the original ER notation [2]. It is supported by the ERRCASE tool, a graphical tool for Eclipse IDE. The EERCASE was also developed using the EMF and GMF. The SQL/DDL generation for PostgreSQL and the model validation were constructed using Epsilon Validation Language and Epsilon Generation Language. The EER metamodel claims to be a technologically independent model, but it only provides concepts for Chen's notation. The GEDBM gathers concepts for several modeling languages. It also provides concepts for some notations, such as MIN-MAX. Besides, the new GEDBM gathers geographical concepts, unlike the EER metamodel. Besides, it does not represent MIN-MAX notation, while the GEDBM does. The EERCASE tool is bounded to Chen's notation and generates SQL code for PostgreSQL. The GEDBM allows code generation for the ANSI/SQL, a standard specification for SQL.

The original GEDBM does not gather concepts for geographical models limiting the metamodel scope. In the revised and extended GEDBM, the OMT-G model concepts are added to the original metamodel to enable geographical modeling. Besides, a new set of M2T rules for SQL/SFS is proposed, expanding the metamodel applicability. Besides, the OMT-G Designer was integrated to guarantee a graphical interface for at least one geographical modeling language.

VI. Conclusion

This paper presented a revised metamodel for relational and geographical database projects, supported by an automatic code generator for ANSI/SQL and SFS/SQL. The metamodel allows projects to use several modeling languages and notations for relational databases. The geographical part was constructed by mapping the OMT-G concepts directly into the extended GEDBM metamodel. The revised GEDBM aims to fit the gap of generic models for database development since there is no generic model that includes concepts from both geographical and relational models. Besides, current metamodels are bound to specific languages and notations, limiting the designer's choice of the most appropriate approach.

It also presented an MDA methodology to assist the database project that allows the integration of tools since the metamodel is generic for extant modeling languages. A set of QVT rules was developed from OMT-G to the extended GEDBM. A set of M2T transformations for ANSI/SQL and SFS/SQL was adapted and reused from the original GEDBM to generate automatic codification. It developed a modeling tool based on Chen's notation, consisting of a set of plug-ins for Eclipse using the EMF for the revised GEDBM, the GMF for the ER's visual concepts, and the Acceleo for both cartridges for code generation. The developed framework provides a substantial difference from other ER tools because it is not bound to a specific language or notation and allows geographical modeling. It is also extensible since it was constructed on top of GEDBM. The code generation uses the ANSI/SQL and the SFS/SQL patterns to overcome the technological dependency of certain DBMS.

For future work, it is necessary to apply OCL rules to guarantee the database normalization process and avoid inconsistent models. The OCL assists in the identification of not well-formed modeling before model-to-model transformations or code generation. A reverse engineering Domain Specific Language is proposed using the Xtext tool, which provides grammar constructions for metamodels.

References

[1] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*. Pearson Addison Wesley, 2015.

[2] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM transactions on database systems (TODS)*, vol. 1, no. 1, pp. 9–36, 1976.

[3] T. Halpin, "Metaschemas for er, orm and uml data models: A comparison," *Journal of Database Management*, vol. 13, no. 2, pp. 20–30, 2002.

[4] T. A. Bruce, *Designing quality databases with IDEF1X information models*. Dorset House Publishing Co., Inc., 1992.

[5] F. Lisboa and C. Iochpe, "Specifying analysis patterns for geographic databases on the basis of a conceptual framework," in *Proceedings of the 7th ACM GIS*, 1999.

[6] F. Lisboa, V. F. Sodré, J. Daltio, M. F. Rodriguês Jr., and V. M. Vilela, "A case tool for geographic database design supporting analysis patterns," in *Proc. of Conceptual Modeling for Advanced Application Domains. 1st Int. Workshop on Conceptual Modelling for GIS*, Springer, 2004.

[7] K. Borges, C. Davis Jr, and A. Laender, "Omt-g: An object-oriented data model for geographic applications," *GeoInformatica*, vol. 5, 09 2001.

[8] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, *MDA Distilled: Principles of Model-Driven Architecture*. Addison Wesley, 2005.

[9] J. Poole, D. Chang, D. Tolbert, and D. Mellor, *Common warehouse metamodel*. John Wiley & Sons, 2002.

[10] OMG, "Mofmodel to text transformation language," 2008.

[11] G. Booch, J. Rumbaugh, and I. Jacobson, *UML: User's Guide*. Campus, 2017.

[12] R. N. Fidalgo, E. Alves, S. España, J. Castro, and O. Pastor, "Metamodeling the enhanced entity-relationship model," *Journal of Information and Data Management*, vol. 4, no. 3, pp. 406–406, 2013.

[13] A. Rosa, I. Gonçalves, and C. E. Pantoja, "A mda approach for database modeling," *Lecture Notes on Software Engineering*, vol. 1, no. 1, pp. 26–30, 2013.

[14] OMG, "Meta object facility (MOF) Query/View/Transfomation specification," 2011.

[15] OMG, "MOF model to text transformation language, v 1.0," 2008.

[16] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *Emf: Eclipse Modeling Framework*. Pearson Education, 2008.

[17] Obeo, "Acceleo: MDA generator." http://www.acceleo.org/, 2023.

[18] L. Lizardo and C. Davis Jr, "Omt-g designer: A web tool for modeling geographic databases in omt-g," pp. 228–233, 10 2014.