IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 May 2023, accepted 10 June 2023, date of publication 19 June 2023, date of current version 27 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3287796

RESEARCH ARTICLE

ER+: A Conceptual Model for Distributed Multilayer Systems

GONÇALO CARVALHO[1], JORGE BERNARDINO[1,2], (Member, IEEE), VASCO PEREIRA[1], AND BRUNO CABRAL[1]

[1]Centre for Informatics and Systems of the University of Coimbra (CISUC), Department of Informatics Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
[2]Polytechnic of Coimbra—Coimbra Institute of Engineering (ISEC), 3045-093 Coimbra, Portugal

Corresponding author: Gonçalo Carvalho (gcarvalho@dei.uc.pt)

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) under PhD grant 2022.12090.BD, and within the scope of the project CISUC-UID/CEC/00326/2020 and by the European Social Fund, through the Regional Operational Program Centro 2020. Also by Project No. 7059 - Neuraspace - AI fights Space Debris, reference C644877546-00000020, supported by the RRP - Recovery and Resilience Plan and the European Next Generation EU Funds, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation.

ABSTRACT Distributed databases and data transformation mechanisms are highly relevant to Business Intelligence and Data Analytics. Most enterprise systems today have multiple and distributed databases, and the growing dissemination of Edge-Oriented architectures is driving this trend across many industries and business domains. The Entity-Relationship (ER) model is fundamental to modeling complex enterprise systems, but it has shortcomings. In particular, the ER model cannot represent data transport between different locations (or databases) of a system, nor can it conceptually express data transformation operations, such as aggregate and line functions, that are standard in data analytics. Therefore, we propose ER+, an extension of the ER model, where data distribution, data transport, data transformation, and information generation for distributed operational and analytical systems can be visually identified. The new ER+ representation has another important benefit, it provides the basis for transforming conceptual models into physical implementations of distributed data-oriented systems. This work introduces new concepts in ER modeling and illustrates their application. The practicality of the ER+ model is also demonstrated through the TPC-H benchmark.

INDEX TERMS Conceptual model, databases, distributed data storage, multilayer systems.

I. INTRODUCTION

Data modeling generates a visual representation of an Information System (IS) or portions of it, to exhibit data objects and their relationships. Simsion and Witt [1] defined data modeling as a "*task consisting of analysis (of business requirements) followed by design (in response to those requirements).*" It is a design activity that classifies information in an organized way and defines its relationships. Conceptual modeling is crucial for modelers who deal with conceptual models, researchers who aim to evaluate and adapt conceptual models, and vendors involved in developing modeling tools [2]. Conceptual modeling is an essential phase in database design, allowing an abstract representation of data without regard to efficiency and implementation constraints. The resulting conceptual schema must be stable and easy for end-users to understand and use [3].

In 1976, Chen [4] introduced the ER model for a unified view of structured data. Through the concept of entities and relationships, the author aims to overcome the semantic uncertainties of information and its structure, and how data is manipulated for storage. Data itself can be of two types: structured and unstructured. Structured data has a well-defined and organized format (e.g. customer information) [5]. Conversely, unstructured data lacks a predefined data model and is unorganized, including emails, photos, and videos. As a result, unstructured data is often more challenging to analyze than structured data because it requires additional processing to extract meaningful information [6]. There are several ER notations for representing data specifications, such as

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

Chen's, Crow's Foot, Barker's, and Unified Modeling Language (UML) through class diagrams.

The fundamental characteristic of the ER model is the strict distinction between data and processes. The ER model is essentially data-oriented, leaving behavioral issues undefined. It is crucial to define the data model to represent the data structure, relationships, and constraints [7]. However, none of the mentioned notations, or their extensions, consider the following concepts. *i)* Data transport (a representation of the flow of data, from one source to another). *ii)* Data transformation (determined by functions for data summarization and/or aggregation to create new data) [8]. *iii)* Data distribution (by setting visual boundaries between data sources and destinations). And *iv)* information generation (selecting appropriate aggregation functions, and grouping data to be used in decision-making processes). Despite the acceptance and use of several conceptual methods and models, their focus on capturing a high level of abstraction is still not sufficient to follow the recent computation evolution [9]. This evolution includes new computing paradigms, such as edge and cloud computing, with different resources and access speeds, communication technologies, and overall data production, distribution, and use.

Therefore, it is essential to enable the representation of the mentioned data dynamics. The real challenge is creating a conceptual model of the entire distributed system and data, coupled with tools expansion to interpret this new model and automatically deploy the system. ER+ is a conceptual model for distributed and-or multilayer systems, and being a conceptual model, it does not include some features that its physical implementation needs to ensure, such as data consistency, data partitioning, replication handling, security, scalability, fault-tolerance, as well as the need to manage concurrent access to shared data. Currently, there are tools and models to design all the components of an IS. UML has categories of diagrams that solve the modeling needs of many aspects of IS. However, it is necessary to evolve these diagrams and tools to incorporate today's data requirements for faster and more accurate decision-making processes. To extract more from them than just the communication and representation of the features of the structures and the functioning of the system [8]. Regardless of the notation, current ER approaches are incapable of such representation.

Object-Oriented Databases (OODBs) could assist in solving the mentioned limitations because OODBs allow modeling complex relationships, focus on managing object-oriented data within a single database instance and storing and retrieving objects directly [10]. However, OODBs are not typically used for creating conceptual representations of systems. ER models, on the other hand, are explicitly designed for conceptual modeling and capturing high-level system concepts. Furthermore, OODBs might not have built-in data transport and transformation features in distributed data systems. Still, developers can address this issue using specialized design languages.

This work aims to answer the following research question: **How can a distributed data architecture be represented using an Extended Entity-Relationship (EER) model, taking into account operational goals, data analytics, data structures, data transport, and data transformation across an entire system?**

This work makes the following contributions:

- Reviews the major developments and extensions of the ER model;
- Proposes an extension to the ER model for distributed multilayer systems, called ER+, which can represent data distribution, data transport, data transformation, and information generation for distributed operational and analytical systems;
- Shows the application of ER+ to the conceptual model of the TPC-H benchmark and specifies the transformation from the ER+ to the relational model.

The remainder of this paper is organized as follows. Section II provides a review of the main developments in ER conceptual modeling approaches. Section III introduces ER+ as a proposed extension to the ER model. In Section IV, we adopt the TPC-H conceptual model and apply ER+ to defined queries to demonstrate its applicability. Finally, Section V concludes the paper and presents some ideas for future work.

II. RELATED WORK

This section reviews data modeling approaches that extended the original ER model to assess if the state of the art in conceptual modeling allows implementing data transformation and transport mechanisms. Such mechanisms are necessary to cope with recent computing paradigms, such as edge and cloud systems, with distributed data sources and where users need valuable information promptly.

In 1976, Chen [4] presented the ER model. Its major advantages are ease of use, a small set of supported constructs, and a clear understanding of the visual representation. Limited constraints and specifications, loss of information content, limited relationship representation, and no representation of data manipulation are some shortcomings of this model. Its central constructs are: i) the entity sets, capturing real-world objects; ii) the relationship sets, capturing associations among objects; and iii) the attributes representing properties of the entity or relationship set.

The Enhanced or EER model is a high-level or conceptual data model incorporating extensions to the original ER model, used in the design of databases [11]. The EER model includes all the key concepts introduced by the ER model and incorporates the concepts of a subclass and superclass. Also, it integrates the notions of specialization and generalization. Furthermore, it introduces the concept of a union type or category.

In 1997, the Object Management Group (OMG) developed the UML [12]. UML provides several diagrams to describe a system from different viewpoints. The class diagram, object diagram, and profile diagram show the static and dynamic

concepts of a system and how they relate to each other. The foundation of many of the UML structures are graphical notations [13]. UML is extensible and has many registered extensions, discussed in this section.

Schiffner and Scheuermann [14] presented a solution for multiple user views or abstractions through an EER model. The concept of abstraction is based on the decomposition of objects hierarchically while preserving a graphical depiction of manageable complexity. The authors also created a Data Definition Language (DDL) that uses the implicit hierarchical structure of entities to achieve concise declarations.

Badia [15] proposed another EER. The authors introduced two concepts. First, they mark each attribute as optional or required. If an entity type has an optional attribute, its entities may or may not have a value of the attribute. If an entity type has a required attribute, its entities must have an attribute's value. In the second, they modeled a choice attribute that can be inclusive or exclusive. If an entity type has an inclusive choice attribute, its entities may have values of one or more attributes in the choice. If an entity type has an exclusive choice attribute, its entities may have only a value of one attribute among the choice.

Thalheim [16] approached the EER after recognizing over 60 extension proposals. The author used the higher-order (or hierarchical) entity-relationship model (HERM). The extended modeling constructs encompassed attributes, entities, clusters, first and higher-order relationships, integrity constraints, and operations. The author also mentioned aggregation functions and classified them into distributive (count, sum, min, and max), algebraic (average and covariance), and holistic (most frequent, rank, and median).

Mani [17] proposed a EER called EReX, with the following extensions: category relationship types, similar to an IS-A relationship but may have an empty key and have an integrity constraint called coverage; they can specify total and exclusive coverage constraints for categories and roles of entity types in relationship types, and they can specify order constraints for participants of a relationship.

Tavana et al. [18] mentioned the concept of entity clustering to tackle some shortcomings of ER diagrams, such as difficulties in understanding and managing large designs. Thus, large ER diagrams can be decomposed into smaller modules by clustering closely related entities and relationships. Teorey et al. [19] developed this concept into grouping as an operation that combines entities and their relationships to form a higher-level construct, which result is called entity cluster. It is a *virtual* or *abstract* entity type used to represent multiple entities and relationships in the ER diagram.

Ordonez et al. [20] dealt with the problem of relational queries in data mining projects that create many temporary tables (static) or views (dynamic), unrepresented as entities in the existing ER model. The authors classified potential database transformations, extended an ER diagram with entities capturing database transformations, and introduced an algorithm that automates the creation of this extended ER model.

Trujillo and Luj-Mora [21] presented an UML approach to accomplish the conceptual modeling of the Extract, Transform, Load (ETL) processes. The authors provided mechanisms, such as aggregation, conversion, filter, join, loader, and merge, for an easy and quick specification of the common operations defined in these ETL processes. These operations can be the integration of distinct data sources, the transformation between source and target attributes, and the generation of surrogate keys. Stated advantages are standardization, ease of use, the functionality of UML, and the seamless integration of the design of the ETL processes. These advantages enable reducing the development time of a Data Warehouse (DW), facilitating data management and administration, and allowing dependency analysis.

Farinha [22] extended UML to increase templates' flexibility. The method allows substitutions among elements of different kinds, such as *cross-kind substitutions*. However, require adequate semantics for the binding relationship. The author proposed such semantics as model transformations that must complement the conventional substitutions made by UML.

Marouane et al. [23] developed a new UML profile to consider the real-time variability in design patterns and to select the pattern elements in its instance. The authors' goal was to surpass some shortcomings of the graphical notations. To achieve this, the authors used UML to define a new package, named *Profile*, extending its syntax and its semantics based on new stereotypes, applying their proposal in class, sequence, and use case diagrams. The authors also include the Object Constraint Language (OCL) to enforce the variation points' consistency.

Daniel et al. [24] introduced UMLto[No]SQL to enable a transparent manipulation of data. The authors tackled the partition of conceptual schemas by mapping each fragment to a different data storage solution. This mapping covers the database queries in diverse and heterogeneous servers.

Plazas et al. [25] introduced a novel conceptual data model based on UML profiles and Model Driven Architecture (MDA) for modeling and implementing IoT-based Business Intelligence applications. Their proposal encompasses sensor systems and real-time and stream databases. The tool for model-to-code implementation considers DWs and ETL processes. The authors also include an OCL for the UML profiles.

Vega et al. [26] presented a multilayer database. However, these layers are not network layers but model layers. In the authors' model, they organized information in layers and distributed it according to the generality of the information. From the classic fields of data (first layer), passing through the specific records of data family types (second layer), and matching the unique information of individual data types (third layer).

As the bibliographic research revealed, the need to develop conceptual models that comprise data transport and transformation in distributed systems, with structured information in distinct layers, from the edge to the core of the network, is still an open issue. The aggregation functions in [14], [16], and [21], and the transformations specifications in [17] and [20], lack the necessary conceptual approach. Although [21] presented a valid approach to represent aggregation functions it was developed for UML class diagrams with the aid of notations, which are yet to be interpreted by a modeling tool, and specifically for ETL processes. Moreover, the research field still neglects the ever-growing need to associate data conceptual modeling with the analytical aspect of data. In addition, only [14], [20], and [25] approached the importance of code generation for their model, through a DDL, a new algorithm for database transformations, and a developed tool, respectively. Finally, none of the reviewed authors applied their model to recent computing paradigms (edge and cloud computing), where data placement and information retrieval are critical.

Despite the evolution and extensions of ER diagrams, they still require a comprehensive representation of data in distributed systems. Current conceptual data models do not consider multilayer designs and their corresponding data transport and transformations. Currently, there is the need to represent dynamic data features in ER models, something that UML achieved to some point, through the different diagrams, but not at a conceptual level. Hence, the need for a visual representation of the additional features of abstraction or transformation functions. Also, with the increasing value of information, designing conceptual models that encompass data analytics is crucial for faster and better decision chains because having information from data is the key feature in the decision-making process.

III. ER+: A CONCEPTUAL MODEL FOR DISTRIBUTED MULTILAYER SYSTEMS

Recent developments in data modeling methods primarily focus on mathematical or logical coding [27]. Consequently, there has been no substantial evolution in the graphical representation of system-wide models and even multilayer approaches. Our proposal, ER+, introduces a visual representation for distributed data models, with data transport, and transformations, including the systems' data structure and functionality in a single diagram. The ER+ model extends the original ER, so it keeps unchanged the concepts of entities and relations and the cardinality elements. The following subsections provide a comprehensive introduction to the concepts and symbols of ER+. And to help understand them better, a demonstration is implemented on how such conceptual elements would be converted into a physical (relational) model. Also, relational algebra is used to describe the data's transformation according to each function, taking the freedom of applying the relational algebra to entities and not tables. In this work, the focus is on applying ER+ to structured data.

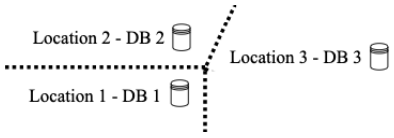


FIGURE 1. The definition of distinct data locations, in this example of three distinct databases.

ER+ extends the ER model with new constructs and is based on the Crow's Foot notation. It enables:

- *Integration of multiple data locations* in the same data model. A location may be a different database, a physical machine, or containers (deployed in the same server or distributed).
- *Functional relationships*, which can represent data transformation and data transport amongst locations (for example, entities or databases), including line functions and group functions.
  - **Data transformation**: from a set of data *S* it produces another set of data *S'* as a result of a function *f* (*S'* can be totally different from *S*, it can have a different number of tuples and/or attributes);
  - **Data transport**: moving a set of data from *A* to *B*, after applying the transformation functions.

In Table 1, we provide a summary of each ER+ concept and corresponding symbol. Figures 2, 3, 5, and 7 represent the full concept of *functional relationships*, which, as previously mentioned, can represent data transformation and data transport among entities, including group definitions and group functions, and line identifiers and line functions. Each individual element is part of the *functional relationship*. In the following examples, *relational algebra* is employed, despite not being meant for entities, but for the relational model. However, to explain ER+ functionalities it will be used on entities, as if they were tables.

A. LOCATIONS DEFINITION

Figure 1 shows a dotted line that works as the border between distinct locations. A location may represent distinct data locations, databases or data repositories, physical nodes, or networks, and each location can have an independent DataBase Management System (DBMS). Thus, allowing the representation of distributed and-or multiplayer systems in a conceptual model.

In this example, there are three distinct locations *Location 1*, *Location 2*, and *Location 3*, each one with an independent database *DB1*, *DB2*, and *DB3*. ER+ does not restrict the possibilities. ER+ defines that when a line (the connection between data sources and destination) crosses the set boundary of the location, it represents data transport. In this representation, the dotted line represents the separation of data locations. The designer is responsible for the number of locations and the borders between those locations, it is not limited to a single border. Just add another dotted line (boundary) to define an additional location, and there is a

TABLE 1. ER+ concepts and symbols.

Name	Summary	Symbol
Locations definition	Sets a boundary between the data source and the data destination after transport. It may represent distinct physical locations, nodes, servers, containers, or a different database in the same server. The number of possible boundaries is only limited by the designer's needs.	
Entity notation add-on	A line will separate all attributes to unambiguously identify the attribute as the source or target of the functional relationship by which contains its start or end. In the case of a *date* type attribute, it may include the time granularity for data.	Entity PK id1 (serial) attribute2 (varchar) attribute3 (int)
Functional relationships	Group definitions (allow applying aggregation functions to subgroups of tuples in a relation).	
	Aggregate functions (uses a set of tuples as an argument and generates a value for each aggregate set). Inside the symbol (half circle), the designer sets the type of function according to the symbols defined in Table 2.	
	Line functions (it is a function that is applied to the linked attributes. The order of the operation is applied by a *top-down* approach, which means that the items on top will be on the left of the operation). Inside the symbol, the designer sets the type of the line function.	
	Inputs for the data operations (used only in the aggregate functions or line functions).	
	Inputs for the data operations (used in the aggregate or line functions, but also become an attribute in the destination entity).	
	Inputs for the group definitions or line identifiers (attributes that will set sub-groups of data).	
	Data transport (moving a set of data from *A* to *B*, after applying the transformation functions. This solid line with an arrow must cross the boundary set by the *Locations definition*).	

visual effect to illustrate a distributed system with several data locations.

B. GROUP DEFINITIONS AND AGGREGATE FUNCTIONS

The group definitions allow applying aggregation functions to subgroups of tuples, where the subgroups are based on attribute values, called the grouping attribute(s) [28]. SQL has a *GROUP BY* clause for this purpose. The grouping attributes aggregate, in the same group, all attributes with the same value as the grouping attribute(s) in the *GROUP BY* clause [5]. In relational algebra, it is represented by $\gamma_{attribute}(Entity)$.

The group definitions are represented by a circle (Figure 2 in red). This element connects the attributes that will set the groups by which data is categorized. The inputs will be the attributes set with a dashed line. Each group will consist of the tuples that have the same value of group attribute(s) [28]. The relationships among entities are used to identify the combination of the attributes' values that will determine each group. From here, one or more dashed lines are connected to a half circle (in red), which is the representation for aggregate or group functions, in this example an *average* function. In addition, the attribute(s) of the group definition

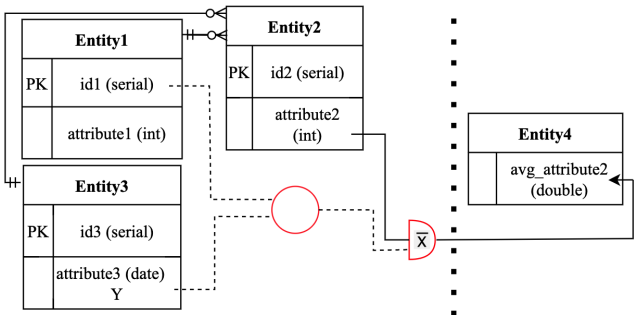


FIGURE 2. Example of a group definition and an aggregate function (average).

will be set as the Primary Key attribute(s) in the destination entity.

An aggregate function uses a set of tuples as an argument and generates a value for each aggregate set [29]. Aggregate functions summarize information about large amounts of data, by representing a set of items through a value or classifying items into groups and determining one value per group [30]. In relational algebra, these aggregate functions are represented as:

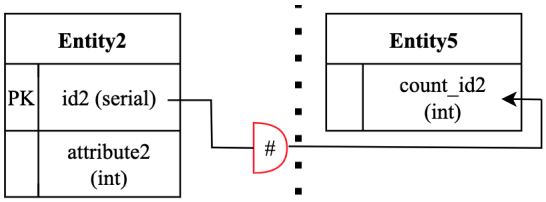


FIGURE 3. Example of an aggregate function (count) without a group definition.

TABLE 2. Functions for data transformation.

Legend	Abbreviation	Symbol
Average	avg	$\bar{x}$
Count	cnt	#
Custom Function	cf	f(x)
Division	div	÷
Maximum	max	M
Minimum	min	m
Multiplication	mul	*
Remove Null Values	rnn	NN-1
Replace Null Values with 0 (zero)	rnn0	NN-0
Subtraction	sub	-
Sum	sum	+

$\gamma_{function(attribute)}(Entity)$, in the case of the function to be a *count* (Figure 3), the attribute is outside the function, thus it is represented as: $\gamma_{attribute,COUNT(*)\rightarrow count\_attribute}(Entity)$

Moreover, these functions may consider more than one aggregate attribute. There are two forms of aggregation, called scalar aggregates and aggregate functions [31]. Scalar aggregates calculate a single scalar value (e.g., the sum of the salaries of all employees). Aggregate functions determine a set of values for each aggregate attribute (e.g., the sum of salaries for each department).

Table 2 displays the data transformation functions. The symbol *f(x)* represents a complex function (*non-primitive*), where *f* will have the parameters, but the user will set the functionality *a posteriori* (the output and group definitions representation and meaning do not change). The symbols are self-explained ($\bar{x}$, #, ÷, *, −, +). ER+ does not limit data transformation functions to these, which are only examples.

Figure 2 shows where the grouping attributes are connected to the group definition and then connected to the group function (half circle). The data aggregation is done based on the relationships between entities of the ER model, in this case, the relationships 1-N between *Entity1* and *Entity2* (Entity1 (1, 1) / Entity2 (0, N)), and *Entity3* and *Entity2* (Entity3 (1, 1) / Entity2 (0, N)). In Figure 2, the *average* represents the average of all entries' values, with links from the group definition attributes (dashed lines) *id1* from *Entity1* and *attribute3* from *Entity3*, and the attribute(s) to use in the aggregate function (solid lines) *attribute2* from *Entity2*. Thus, ER+ will calculate the average of *attribute2* from *Entity2* and will group the results by *id1* from *Entity1* and *attribute3* from *Entity3*, which in this case is a date type. This operation can be represented with relational algebra as:

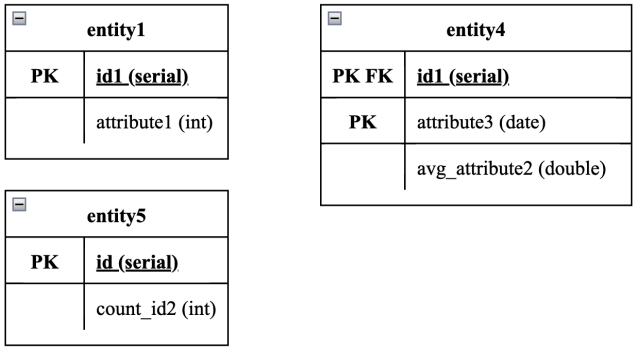


FIGURE 4. Physical model of the conceptual model of Figures 2 and 3.

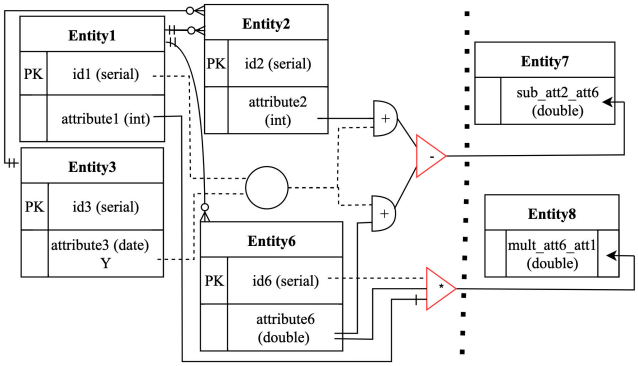


FIGURE 5. Example of line functions that are represented by a triangle.

$$\gamma_{id1,attribute3,AVG(attribute2)\rightarrow avg\_attribute2}(Entity1 \bowtie Entity3 \bowtie Entity2)$$

In Figure 3, a *count* of the attribute, will show the number of entries of that attribute, and because it does not have a group definition, the result is a scalar value. This aggregate function without a group definition *id2* from *Entity2* will result in *Entity5* with the attribute *count_id2*. The output of the aggregate functions can be a line function, or the new data can be directly transported to the new entity in a distinct location.

To further explain this concept, it is displayed the conversion of ER+ into the relational model and an analysis of the operations in Figures 2 and 3. In Figure 2, the location on the right of the vertical dashed line will materialize a table called *entity4* with the Primary Key <*id1, attribute3*> and the attribute to store the average value *avg_attribute2*. Also, a table *entity1* with the same attributes (*id1* and *attribute1*). In the proposed conceptual model of Figure 3, ER+ will set the generation of a new table *entity5*, with the attribute *count_id2* to store the result of the function.

Thus, if the attribute that sets the group definition is a Primary Key in the physical model, the entity that contains that attribute will also become a table in the destination. That attribute will be a Primary Key and Foreign Key in the new table and the Primary Key in the solution used for the data from the source table. Otherwise, if an attribute in a

group definition or line identifier is not a Primary Key, the attribute will only become the Primary Key in the new table, thus, without the source table data. Figure 4 displays the physical model created by the ER+ conceptual examples of Figures 2 and 3.

C. LINE FUNCTIONS

The line functions are represented with a triangle (Figure 5 in red). The definition of the function itself is then introduced inside the triangle. These can be of two types: with a group definition (Figure 5 top), and without a group definition but a line identifier (Figure 5 bottom). The order of the operation is applied by a *top-down* approach, which means that the items on top will be on the left of the operation. The inputs for the entity, in the new location, are the result of the aggregate functions.

Using the examples provided with Figure 5, on the top, there are two attributes that will be aggregated by a *sum* function (*attribute2* from *Entity2* and *attribute6* from *Entity6*) and grouped by two other attributes (*id1* from *Entity1* and *attribute3* from *Entity3*). After this aggregation, the new values are going to be used in the line function *subtraction*. In the example on the bottom, there is a 1-N relationship between *Entity1* and *Entity4*, and there is no group definition, so the attributes marked with the solid line (*attribute1* from *Entity1* and *attribute6* from *Entity6*) will be *multiplied* and will have a line identifier (dashed line), the *id6* from *Entity6*. In addition, the attribute with the solid line without a *vertical line* near the triangle will be used only in the line function, and the attribute with the solid line with a *vertical line* near the triangle will, automatically, also become an attribute in the destination entity. Using relational algebra to assist with the details of the concept, this is the same as a projection, represented as $\pi_{attribute}(Entity)$, and more specifically in this example:

$$\pi_{(query1 \rightarrow sum\_att2 - query2 \rightarrow sum\_att6) \rightarrow sub\_att2\_att6}$$
$$(Entity1 \bowtie Entity3 \bowtie Entity2 \bowtie Entity4)$$
$$\pi_{(attribute4 * attribute1) \rightarrow mult\_att6\_att1}(Entity1 \bowtie Entity4)$$

The output of the line functions will be new data to store in the destination location. When converting the conceptual model to a physical model, in this example, the ER+ will create new tables in the destination layer. The *subtraction* will create a table (*entity7*) with three fields, a composite Primary Key with *id1* from *Entity1* and *attribute3* from *Entity3*, and the data field after the functions *sub_att2_att6*, and will also create a projection (*foreign table*) of *Entity1* because its Primary Key is used in the group definition (as explained before).

In the bottom case, a new table will encompass three fields, the Primary Key *id6* from *Entity6*, the result of the operation *mult_att6_att1*, and *attribute1* as a copy (projection) from *Entity1*, because this attribute has a *vertical line* near the line function. And ER+ will also create a new table (*Entity6*) in this layer (as a foreign table or an exact copy) with the same attributes (*id6* and *attribute6*).

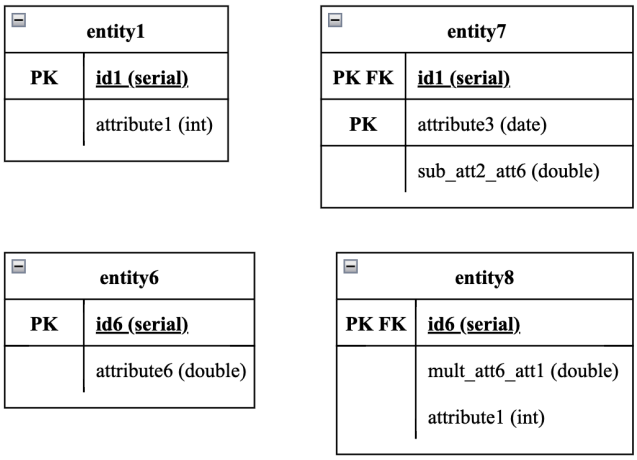


FIGURE 6. Physical model of the conceptual model of Figures 2 and 3.

Figure 6 shows the physical model of the ER+ representation of the data transformation and transport presented in Figure 5.

D. INPUTS

In Figure 7, with a solid line (in red), the source attribute(s) (*attribute2* from *Entity2* and *attribute6* from *Entity6* in two different operations) will be the inputs for the data operations, such as the aggregate or line functions. In relation algebra, the selection operator is set by $\sigma_{attribute}(Entity)$.

Identical to the previous concept, in Figure 7, with a solid line (in dark blue) are the source attribute(s) (*attribute1* from *Entity1*) that will be the input(s) for the data operations. The source attributes with a *vertical line* "|", near group definitions or line functions symbols (discussed in III-B and III-C respectively), will, automatically, also become an attribute in the destination entity.

In Figure 7, with a dashed line are the attributes that will set the group definitions (in green) or be the line identifiers (in purple). These attributes will be crucial to creating subgroups of data. In Figure 7 there are two inputs for group definitions (*id1* from *Entity1* and *attribute3* from *Entity3*), and *id6* from *Entity6* has a line identifier. When more than one input is used as a grouping attribute, these must connect to a circle, in order to enable a direct visualization and allow the use of the same group definition in several aggregate functions. Thus, two of them *id1* from *Entity1* and *attribute3* from *Entity3* are linked to a circle, that set the group definition. However, when a single input is the grouping attribute it may link directly to the function(s), as in the case of *id6* from *Entity6*, which is known as a line identifier.

As mentioned, if the attribute that sets the group definition or line identifier is a Primary Key, in the physical model, the entity that contains that attribute will also become a table in the destination, and that attribute will be a Primary Key and Foreign Key in the new table, and Primary Key in the copy of the source table. If the attribute in a group definition or line

identifier is not a Primary Key, it will only be the Primary Key in the new table, thus, without the 'copy' of the source table.

E. DATA TRANSPORT

Figure 7 displays the data transport (moving a set of data from *DB1* to *DB2*, after applying the transformation functions) through an output which is represented by a solid line (in gray), starting in an aggregate or line function and ending in an arrow. This transport will culminate in an attribute of an entity, to which the arrow will point. This entity stores the data from *functional relationships* (the result of the group definitions and the line identifiers, and the aggregate and line functions). In addition, a solid line that crosses any boundary, set by a dotted line, represents data transport. In this example, the attributes *sub_att2_att6* and *mult_attr6_att1* (previously explained) will store the transformed data after the transport process. When converting the conceptual model to a physical model, the data transport will create a new table(s) and field(s), depending on the type of operation defined in the source layer.

F. ENTITY NOTATION ADD-ON

In Figure 7, ER+ displays a change to the entity notation by adding a line to separate all attributes (in aqua/light blue). This line will separate all attributes to unambiguously determine the attribute at the source or target of the functional relationship, used in the data transformation process. In the *Crow's Foot* notation, only the Primary Key is separated by a line from the rest of the attributes, but we consider a line to distinguish all attributes, thus, having a clear notion of where a line (solid or dashed) is originated from.

In the case of the attribute being of *date* type, the user has two options: specify the *date* granularity (in this example, *attribute3* from *Entity3* is set to year (Y)) or leave it blank, but still referencing a *date* attribute in the group definition. In the first case, the user defines the granularity amongst Year (Y), Month (M), Day (D), Hour (h), Minute (m), and Second (s). Regarding the latter, ER+ will set, by default, the *date* granularity to its minimum (Second). Setting a granularity is important when a *date* type field is used in a group function because the aggregation function will have distinct results concerning the time granularity. For example, retrieving sales amount *sum* per year, per month, or per minute will display significant differences.

G. THE OUTCOME OF DATA TRANSPORT AND TRANSFORMATION

In Figure 7, a new entity, such as *Entity7* or *Entity8*, demonstrates the outcome of the data transformation and transport. The resulting data from the *functional relationship*, sets an attribute to store the transformed data in a new entity, with the same novel notation add-on. E.g., the attributes *sub_att2_att6* and *mult_attr6_att1* will store the transformed data after the transport process.

When converting the conceptual model to a physical model, the code will include creating the table(s), and attributes, and inserting data into the table(s). The developer has the opportunity to choose the type of 'assistant' tables for the new tables, which all depends on the performance issues that concern the developer. Depending on the transformation and transport (a new location in the same server, or in a distributed location), additional scripts may be required.

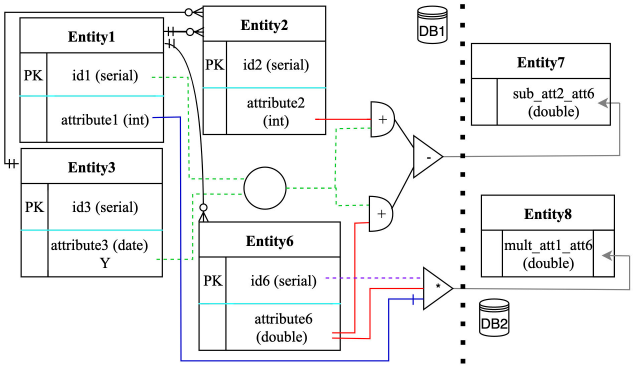


FIGURE 7. Example of the inputs for the operations and the identifier add-on.

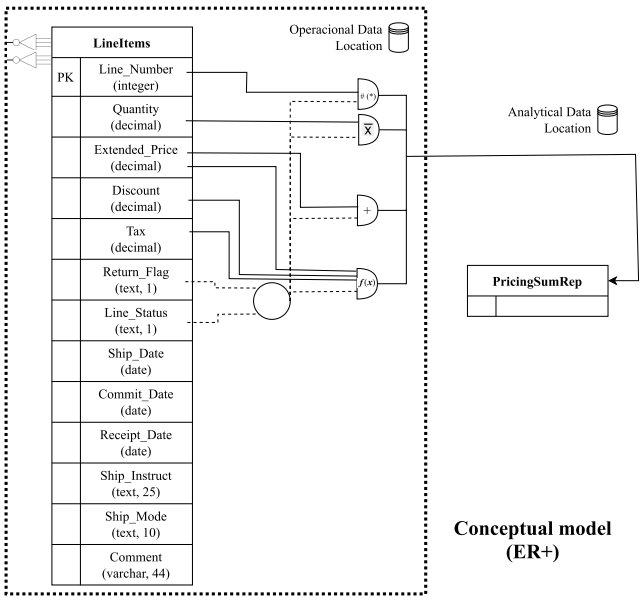


FIGURE 8. ER+ variations.

H. REPRESENTATION VARIATION

ER+ allows several operations and a developer may want to use the same model to introduce many data transformations, ER+ allows for a different representation, to minimize the number of lines crossing boundaries. This subsection introduces a different representation opportunity. Figure 8 displays a single line from different functions. ER+ is responsible to conduct each function in the *Operational Data Location* to the entity in the *Analytical Data Location*. The attributes names will be automatically generated, as in most ER modeling tools that convert the attributes to Foreign Keys.

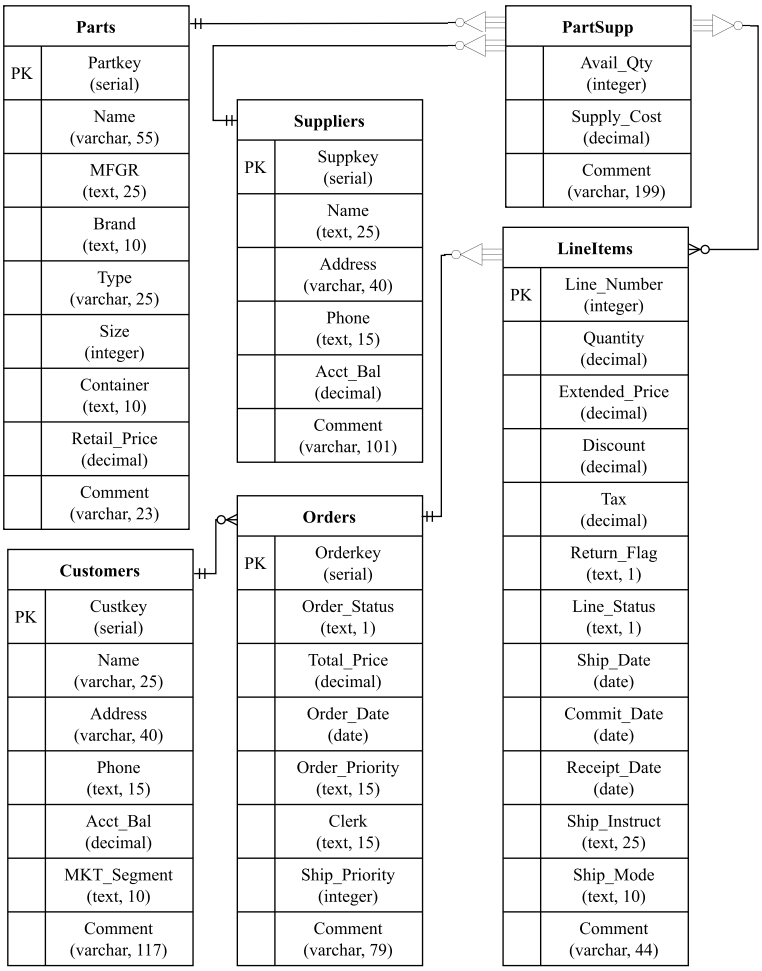


FIGURE 9. Conceptual model for the TPC-H use case.

This notation will be employed in section IV, and section IV-A has the physical model of Figure 8 where the attributes of the data transformation and transport are displayed.

I. FUNCTIONAL RELATIONSHIPS SUMMARY

This subsection summarizes the uses for *functional relationships*, which are:

- *Data transport with group definitions and data transformation (Figure 2):* it is possible to set group definitions and aggregate functions. In group definitions, the user sets the granularity of data. The aggregate functions, such as average, count, maximum, minimum, standard deviation, sum, and variance will aggregate the attributes' values. For example, Figure 3 shows a *count* function. In addition, the attribute(s) used in the group definition will be the Primary Key(s) in the destination entity.
- *Data transport without group definitions and with data transformation (Figures 3 and 5):* in these cases, there is not a group definition. There are two possibilities for data transformation:
  - combine lines, such as concatenate, subtract, and multiply attribute values. For example, Figure 5 (bottom) displays a line function (*multiplication*) between the two attributes, in a *top-down* order (despite not having importance in this particular operation), but with a line identifier (the attribute with a dashed line). Also, the solid line with a vertical line near the triangle will represent a new attribute after the transformation.
  - set a group function for all values of an attribute. For example, Figure 3 only has a group function *count*, and this will store in the new entity the *count* of all the entity's attributes, creating a single instance. Because no group definition is set, and the destination entity needs a Primary Key, this will be set automatically, and the attribute will be set as *"id"*.

J. PROOF OF PERFORMANCE

All the data transformations introduced in our study used the logic and rules of relational algebra, which can be

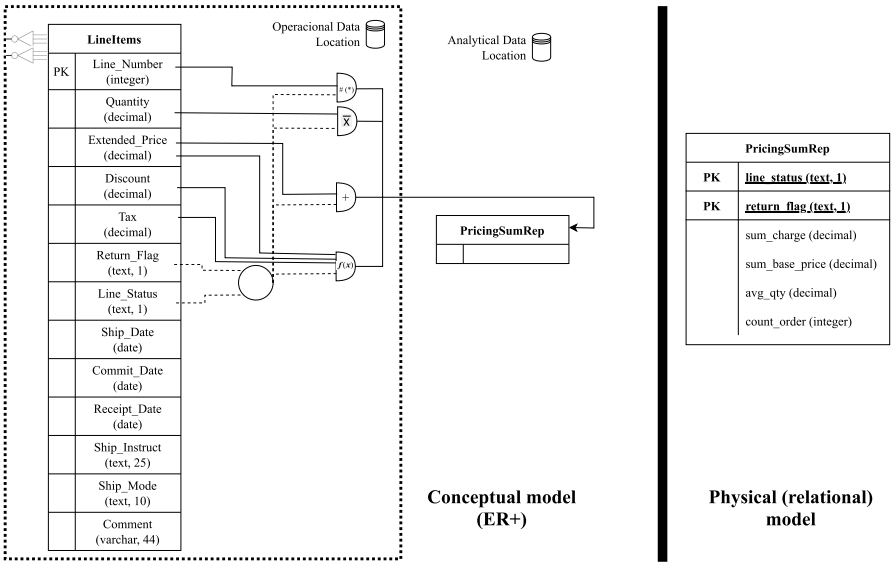


FIGURE 10. Conceptual Model with ER+ of Query 1 (left) and the result of the conversion to the physical model (right).

translated into SQL. Proof of equivalence between SQL and relational algebra can be found in previous works by Guagliardo et al. [32].

Data transport model features were not provided with a relational algebra equivalent, as they only imply data extraction and loading, with no associated transformations.

IV. ER+ APPLIED TO THE TPC-H CONCEPTUAL MODEL

The primary goal of this work is to propose a new extension for modeling the structure and data transformation and transport mechanisms for distributed data systems. This section introduces the conceptual model of the TPC-H benchmark [33] and its queries, to demonstrate the ER+ capabilities and display the transformations to the relational model. In generating the physical model, the conceptual model, ER+, will also produce code for the concepts of data transport and transformation functions.

Figure 9 illustrates the TPC-H conceptual model, which was created from the logical model presented in the TPC-H. This model is applied to highlight the use of ER+ in any scenario, and not only in a handcrafted example, but the entities *Nation* and *Region* were removed. Thus, there are six entities to represent a business. In this business, *Customers* place *Orders* that comprise *LineItems*, to which *Parts* are provided by *Suppliers*. Because there is a *ternary* relationship amongst *Parts*, *Suppliers*, and *Line Items*, there is also an entity with extra attributes called *PartSupp*. In each of the following subsections, there is a discussion on the ER+ conceptual approach to the queries provided in the TPC-H benchmark. In this paper, we visually implement a query through ER+. Thus, we are not going to assess the performance of our model. ER+ allows bypassing queries with data transformation and transport, because the result of the queries, that were modeled with ER+ in the conceptual model, is stored in the new tables, after the physical model deployment. From the original 22 queries, four were selected: Q1, Q9, Q11, and Q22. Because each original query has over one of the same aggregation functions, the queries were simplified to provide a better illustration of ER+ due to space limitations. There is also the representation of the data transformation in a relational model. The names of entities and attributes are identical to the TPC-H benchmark.

A. PRICING SUMMARY REPORT QUERY (Q1)

The Pricing Summary Report Query provides a summary pricing report for all *LineItems* shipped on a given date (the specificity of the date is out of the scope of this work). The original query lists the total for the extended price, discounted extended price, discounted extended price plus tax, average quantity, average extended price, and average discount. It sets the group definition with the attributes *Return_Flag* and *Line_Status*. There is also a *count* of the number of *LineItems* in each group. Because many of the aggregation functions were the same, we used one example of each (*sum*, *average*, *count*, and *function*).

In the conceptual model side of Figure 10, in the *Operational Data Location* there is a group definition with the attributes *Return_Flag* and *Line_Status* with a dashed line to the circle, and four aggregate functions. The first function is a *count* all, represented by the symbol '# (*)' inside the half circle, the solid line departs from the entity Primary Key attribute (*Line_Number*). The second function, in the *Operational Data Location*, is an *average*, represented by the symbol '$\bar{x}$' inside the half circle, of the attribute *Quantity*. The third function is a *sum* of the *Extended_Price* attribute, represented by the symbol '+' inside the half circle. Finally, the fourth, in the *Operational Data Location*, is a *custom function*, represented by the symbol 'f(x)' inside the half

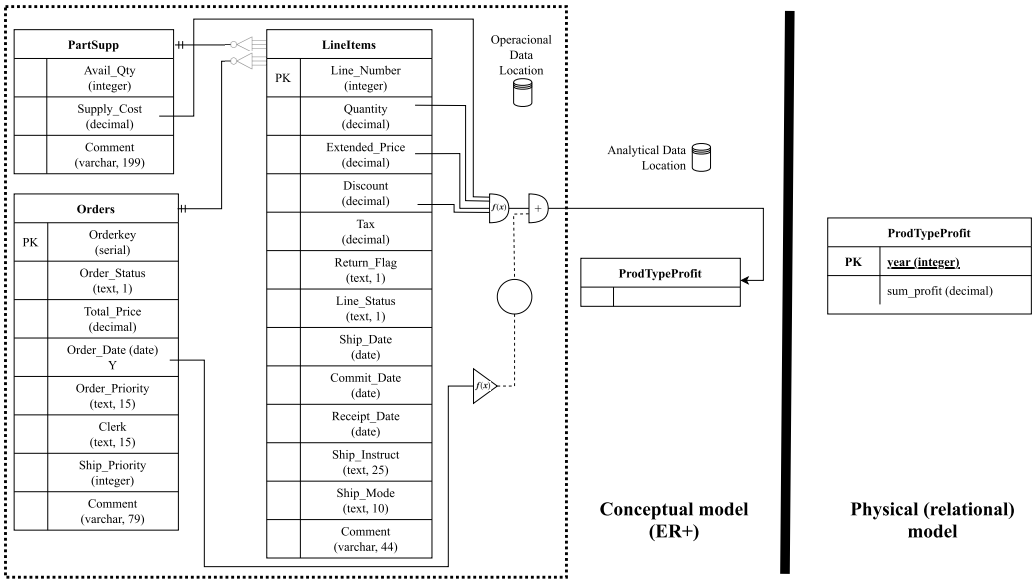


FIGURE 11. Conceptual Model with ER+ of Query 9 (left) and the result of the conversion to the physical model (right).

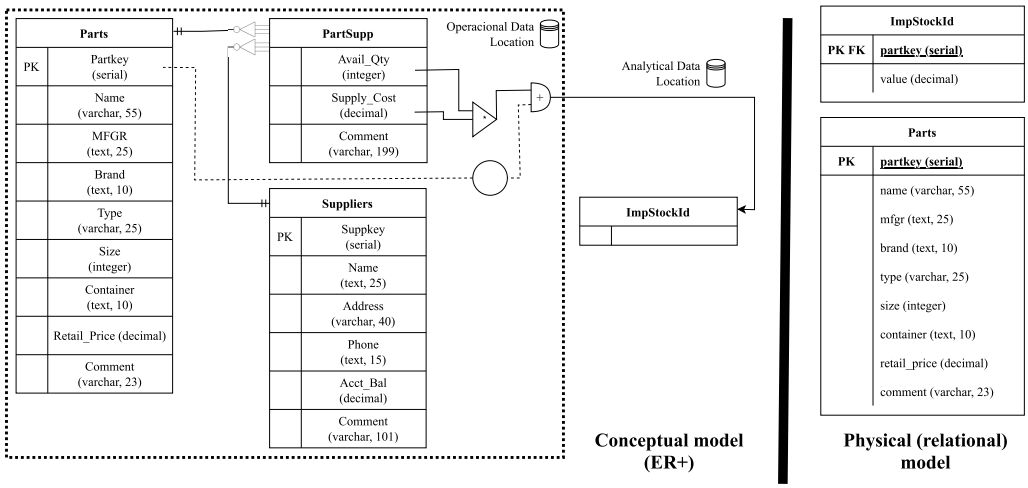


FIGURE 12. Conceptual Model with ER+ of Query 11 (left) and the result of the conversion to the physical model (right).

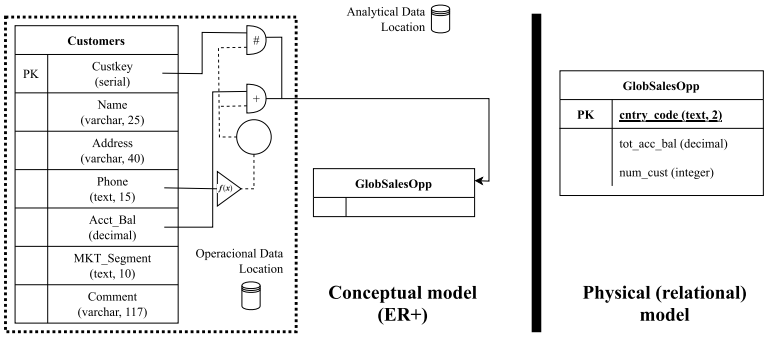


FIGURE 13. Conceptual Model with ER+ of Query 22 (left) and the result of the conversion to the physical model (right).

circle, so the designer will have to define, in the code, the components of the function. This function has three attributes *Extended_Price*, *Discount*, and *Tax* which will be used according to the Equation 1. The results of these

functions will be transported to the entity *PrincingSumRep* in the *Analytical Data Location* layer.

$$SUM(Extended\_Price \times (1 - Discount) \times (1 + Tax)) \quad (1)$$

In the physical model side of Figure 10, there is the corresponding physical model of Q1 after the conversion of ER+ from the conceptual model. The table is called *PrincingSumRep*, with two attributes as Primary Keys, resulting from the group definition attributes (*Return_Flag* and *Line_Status*), and four additional attributes due to the aggregate functions. The attributes *count_Order* (as defined in the benchmark), *avg_qty*, *sum_base_price*, and *sum_charge* store the mentioned functions *count*, *average*, *sum*, and *custom function* respectively.

B. PRODUCT TYPE PROFIT MEASURE QUERY (Q9)

The Product Type Profit Measure Query finds, for each year, the profit for all parts ordered in that year. The profit is defined in Equation 2 for all *LineItems* describing parts in the specified line. In the conceptual model side of Figure 11 (left), in the *Operational Data Location*, there is a *custom* line function, represented by the symbol 'f(x)' inside the triangle, to *extract* the year of the attribute *Order_Date*, as set by the letter "Y" in the third column of the entity *Orders*. The result will be the group definition for the aggregate function *sum*, represented by the symbol '+' inside the half circle, that will group the result of another *custom* function as defined in Equation 2, represented by the symbol 'f(x)' inside the half circle. In the *Analytical Data Location*, the entity *ProdTypeProfit* will store the results of the mentioned functions. The *Nation* entity was not included from the original model, so this query does not encompass the selection of the *nation* attribute, present in the original query.

$$SUM(Extended\_Price \times (1 - Discount) - (Supply\_Cost \times Quantity)) \quad (2)$$

C. IMPORTANT STOCK IDENTIFICATION QUERY (Q11)

In the physical model side of Figure 11 (right), the table *ProdTypeProfit* stores the data set in the entity with the same name in the *Analytical Data Location* layer. There is a line function *extract* applied to the attribute *Order_Date*, that set the group definition, so ER+ originates the Primary Key (*year*) in the destination table. Also, the field *sum_profit* stores the result of the aggregate functions after transport.

The Important Stock Identification Query finds, from scanning the available stock of suppliers, all the parts that represent a significant percentage of the total value of all available parts (the specificity of the percentage is out of the scope of this work). The query displays the part number and the value of those parts. In the conceptual model side of Figure 12 (left), in the *Operational Data Location*, there is a group definition set by the Primary Key (*Partkey*) of the entity *Parts*, and a line function that will multiply the attributes *Avail_Qty* and *Supply_Cost*, represented by the symbol '*' inside the triangle. ER+ will use the result as input for an aggregate function *sum*, represented by the symbol '+' inside the half circle, with the group definition *Partkey*. Thus, in the *Analytical Data Location*, the entity *ImpStockId* will store the result of this operation.

The conversion of the ER+ conceptual model to the corresponding physical model is illustrated in Figure 12 (right). The group definer is a Primary Key, so ER+ will automatically create a table, as a copy, with the same name and Primary Key. This key will be a Foreign Key in the new table (*ImpStockId*), which will store the result of the line and aggregate functions in the attribute *value* (as defined in the benchmark).

D. GLOBAL SALES OPPORTUNITY QUERY (Q22)

This query counts how many customers within a specific range of country codes have not placed orders for seven years but who have a greater than average "positive" account balance (the specificity of the time and average clauses are out of the scope of this work). The *country-code* is defined as the first two characters of the attribute *Phone*. In the conceptual model side of Figure 13, in the *Operational Data Location*, there is a line function *substring*, represented by the symbol 'f(x)' inside the triangle, to get the first two characters of the attribute *Phone*, which will be used as the group definition. The aggregate functions are a *count* of every customer in that group (country code), represented by the symbol '#' inside the half circle, and the *sum* of the *Acc_Bal*, represented by the symbol '+' inside the other half circle. These two aggregate functions will be represented in the *Analytical Data Location* in a new entity, *GlobSalesOpp*.

In the physical model side of Figure 13, the Primary Key is set with the group definition attribute *Phone*, that after the line function *substring* originated the key *cntry_code*. And the two fields, *num_cust* and *tot_acc_bal* store the result of the aggregate functions after transport.

V. CONCLUSION AND FUTURE WORK

We introduced ER+ to answer our research question: **How can a distributed data architecture be represented using an EER model, taking into account operational goals, data analytics, data structures, data transport, and data transformation across an entire system?** We specified ER+, a new conceptual model that can visually represent a distributed and-or multilayer system with one or more locations (each location can have an independent database engine of different types). This model supports functionalities for data transformations (line and group functions) and their transport to a new storage location. ER+ also provides the necessary concepts for future data modeling tools to span multiple locations and code generation for automatic deployments. This enables DevOps teams to leverage development by enabling a holistic process, from conceptual design to deployment, reducing time and effort for both development and deployment.

In addition to the new ideas developed for conceptual modeling, we also aim to create the basis for *i)* future development

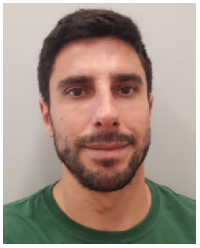
of building and deploying a complete distributed database system from its conceptual model using automatic code generation approaches; *ii)* updating existing databases with the aforementioned features. The use case illustrates all the introduced concepts of multilocation databases and *functional relationships*. ER+ introduces conceptual modeling, data manipulation, transformation, and transport capabilities to help understand the data model and enable data analysis and business intelligence capabilities.

The future work includes implementing ER+ with unstructured data and developing a modeling tool that implements the ER+ concepts of data transformation and transport. The main features of distributed systems, discussed in Section I, will be included in its development, and we will investigate methods for incorporating these characteristics into the conceptual model.

Our approach, which supports the DevOps development process, takes into account all the ER+ features introduced and enables a holistic view from design to deployment.

REFERENCES

[1] G. Simsion and G. Witt, *Data Modeling Essentials*. Amsterdam, The Netherlands: Elsevier, 2004.
[2] D. Bork, D. Karagiannis, and B. Pittl, "Systematic analysis and evaluation of visual conceptual modeling language notations," in *Proc. 12th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, May 2018, pp. 1–11.
[3] J. Akoka and I. Comyn-Wattiau, "Entity-relationship and object-oriented model automatic clustering," *Data Knowl. Eng.*, vol. 20, no. 2, pp. 87–117, Oct. 1996.
[4] P. P.-S. Chen, "The entity-relationship model-toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, 1976.
[5] Y. Bai and S. Bhalla, *Introduction to Databases*. Reading, MA, USA: Addison-Wesley, 2022.
[6] M. Ahmed, "Data summarization: A survey," *Knowl. Inf. Syst.*, vol. 58, no. 2, pp. 249–273, Feb. 2019.
[7] H. Aljumaily, D. Cuadra, and D. F. Laefer, "An empirical study to evaluate students' conceptual modeling skills using UML," *Comput. Sci. Educ.*, vol. 29, no. 4, pp. 407–427, Oct. 2019.
[8] G. Carvalho, J. Bernardino, V. Pereira, and B. Cabral, "A holistic data modeling approach for multi-database systems," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 5865–5867.
[9] K.-D. Schewe, K. Bósa, A. Buga, and S. T. Nemes, "Conceptual modelling of service-oriented software systems," *Int. J. Conceptual Model.*, vol. 13, pp. 216–233, Feb. 2018.
[10] M. Zand, V. Collins, and D. Caviness, "A survey of current object-oriented databases," *ACM SIGMIS Database, DATABASE Adv. Inf. Syst.*, vol. 26, no. 1, pp. 14–29, Feb. 1995.
[11] S. Bagui, *Extended Entity Relationship Modeling, in Encyclopedia of Database Technologies and Applications*. Calgary, AB, Canada: Idea Group Inc. (IGI), 2006.
[12] *OMG Unified Modeling Language (OMG UML)*, OMG, Needham, MA, USA, Dec. 2017.
[13] M. M. Mkhinini, O. Labbani-Narsis, and C. Nicolle, "Combining UML and ontology: An exploratory survey," *Comput. Sci. Rev.*, vol. 35, Feb. 2020, Art. no. 100223.
[14] G. Schiffner and P. Scheuermann, "Multiple views and abstractions with an extended-entity-relationship model," *Comput. Lang.*, vol. 4, nos. 3–4, pp. 139–154, Jan. 1979.
[15] A. Badia, "Conceptual modeling for semistructured data," in *Proc. 3rd Int. Conf. Web Inf. Syst. Eng. (Workshops)*, Dec. 2002, pp. 170–177.
[16] B. Thalheim, *The Enhanced Entity-Relationship Model*. New York, NY, USA: Springer-Verlag, 2007.
[17] M. Mani, "EReX: A conceptual model for XML," in *Proc. 2nd Int. XML Database Symp. (XSym)*, 2004, pp. 128–142.
[18] M. Tavana, P. Joglekar, and M. A. Redmond, "An automated entity–relationship clustering algorithm for conceptual database design," *Inf. Syst.*, vol. 32, no. 5, pp. 773–792, Jul. 2007.
[19] T. Teorey, S. Lightstone, T. Nadeau, and H. V. Jagadish, "4—Requirements analysis and conceptual data modeling," in *Database Modeling and Design*, 5th ed., T. Teorey, S. Lightstone, T. Nadeau, and H. V. Jagadish, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2011, pp. 55–84.
[20] C. Ordonez, S. Maabout, D. S. Matusevich, and W. Cabrera, "Extending ER models to capture database transformations to build data sets for data mining," *Data Knowl. Eng.*, vol. 89, pp. 38–54, Jan. 2014.
[21] J. Trujillo and S. Luján-Mora, "A UML based approach for modeling ETL processes in data warehouses," in *Proc. Int. Conf. Conceptual Modeling*, vol. 2813, 2003, pp. 307–320.
[22] J. Farinha, "Extending UML templates towards flexibility," in *Proc. CEUR Workshop*, vol. 1694, 2016, pp. 32–41.
[23] H. Marouane, C. Duvallet, A. Makni, R. Bouaziz, and B. Sadeg, "An UML profile for representing real-time design patterns," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 30, no. 4, pp. 478–497, Oct. 2018.
[24] G. Daniel, A. Gómez, and J. Cabot, "UMLto[No]SQL: Mapping conceptual schemas to heterogeneous datastores," in *Proc. 13th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, May 2019, pp. 1–13.
[25] J. E. Plazas, S. Bimonte, C. de Vaulx, M. Schneider, Q. Nguyen, J. Chanet, H. Shi, K. M. Hou, and J. Carlos Corrales, "A conceptual data model and its automatic implementation for IoT-based business intelligence applications," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10719–10732, Oct. 2020.
[26] J. Vega, C. Crémy, E. Sánchez, A. Portas, J. A. Fábregas, and R. Herrera, "Data management in the TJ-II multi-layer database," *Fusion Eng. Des.*, vol. 48, nos. 1–2, pp. 69–75, Aug. 2000.
[27] G. Guizzardi, C. M. Fonseca, J. P. A. Almeida, T. P. Sales, A. B. Benevides, and D. Porello, "Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support," *Data Knowl. Eng.*, vol. 134, Jul. 2021, Art. no. 101891.
[28] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY, USA: McGraw-Hill, 1991.
[29] A. Klug, "Equivalence of relational algebra and relational calculus query languages having aggregate functions," *J. ACM*, vol. 29, no. 3, pp. 699–717, Jul. 1982.
[30] G. Graefe, "Techniques for large databases,," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 73–169, 1993.
[31] R. Epstein, "Techniques for processing of aggregates in relational database systems," Dept., Univ. California, EECS, Berkeley, CA, USA, Tech. Rep. UCB/ERL M79/8, Feb. 1979.
[32] P. Guagliardo and L. Libkin, "A formal semantics of SQL queries, its validation, and applications," *Proc. VLDB Endowment*, vol. 11, no. 1, pp. 27–39, Sep. 2017.
[33] *TPC benchmark H—Standard Specification Revision 3.0.1*, Transaction Processing Performance Council, Apr. 2022.



GONÇALO CARVALHO received the B.Sc. degree in geography from the University of Coimbra (UC), in 2005, the M.Sc. degree in geographical information systems from the University of Trás-os-Montes e Alto Douro (UTAD), in 2009, and the B.Sc. degree in informatics engineering from the Polytechnic of Coimbra (IPC), Portugal, in 2016. He is currently pursuing the Ph.D. degree with the University of Coimbra. After his experience, he was a web and software developer, between 2015 and 2018. His major research interests include databases, distributed systems, edge computing, cyber-physical systems, machine learning, and green computing.

JORGE BERNARDINO (Member, IEEE) received the Ph.D. degree from the University of Coimbra, in 2002. From 2005 to 2010, he was the President of the Coimbra Engineering Institute (ISEC). From 2017 to 2019, he was also the President of ISEC Scientific Council. In 2014, he was a Visiting Professor with CMU. He was the Director of the Applied Research Institute (i2A), Polytechnic of Coimbra, from 2019 to 2021. He is currently a Coordinator Professor with the Polytechnic of Coimbra—ISEC, Portugal. He has authored more than 200 publications in refereed conferences and journals and participated in several national and international projects. His main research interests include big data, NoSQL, data warehousing, dependability, and software engineering.

VASCO PEREIRA received the Ph.D. degree in informatics engineering from the University of Coimbra (UC), in 2016. He is currently an Assistant Professor with the Department of Informatics Engineering and a Senior Researcher with the Centre for Informatics and Systems of the UC, where he has been involved in multiple national and European research projects. His main research interests include computer networks, network security, QoS, edge computing, and the IoT. For more information visit the link (http://www.uc.pt/go/vasco).

BRUNO CABRAL received the dual master's degree in software engineering from the Faculty of Engineering and the Ph.D. degree (Hons.) in informatics engineering from the University of Coimbra (UC), in 2009. He was a tenured Professor with the Informatics Engineering Department, UC. He has been an Adjunct Associate Teaching Professor with CMU. He is currently the Coordinator of the Master Program in Software Engineering, UC, and a Senior Researcher with the Systems and Software Engineering Group, Centre for Informatics and Systems of the UC (CISUC). His main research interests include distributed systems, parallel programming languages, machine learning, and dependable computing. He was either the PI or a staff member on multiple EU, government, and privately funded research projects and frequently works as a consultant to the software industry.