Entity/Relationship Graphs

Unifying Data Modelling and Taking Data Integrity Management to the Next Level

Author names surpressed due to double-blind reviewing

ABSTRACT

Chen's Entity/Relationship (E/R) framework is a lingua franca for well-designed database schemata. We define E/R graphs as property graphs that are instances of E/R diagrams. As the latter are (hierarchical) property graphs themselves, and E/R graphs define homomorphisms on E/R diagrams, the entire E/R framework becomes available for modeling hierarchical property graphs. In addition, E/R graphs provide the first graph semantics for E/R diagrams. Further to the unification of conceptual and graph data modeling, E/R graphs have a strong impact on data integrity management. In relational databases, referential integrity is managed by redundantly repeating the key attributes of the referenced table as foreign key attributes in the referencing table. This attribute redundancy is a bottleneck for referential integrity maintenance. However, the use of E/R graphs makes it possible to manage referential integrity by simply utilizing direct edges, called E/R links, between objects at the instance level. As a consequence, data redundancy can be completely eliminated, minimizing sources of potential inconsistency and the effort of update maintenance. This is achieved by E/R keys that combine the use of properties and E/R links to enforce entity integrity, in contrast to property keys that rely exclusively on the use of properties to enforce entity integrity. E/R graphs provide designers with a choice: introduce property redundancy on nodes to manage entity integrity by property keys and referential integrity by foreign keys; or utilize E/R keys for entity integrity and E/R links for referential integrity. Throughout the article, we illustrate all ideas on the schema and instances of the TPC-H benchmark as our running example, but also use extensive experiments with the benchmark to quantify the range of effort associated with i) property keys and E/R keys for managing entity integrity, and ii) property redundancy and E/R links for managing referential integrity. In summary, E/R diagrams form a principled core for well-designed property graphs within the recently proposed class of PG-schemata, while E/R keys constitute a computationally-friendly core of the recently proposed class of PG-keys.

ACM Reference Format:
Author names surpressed due to double-blind reviewing. 2025. Entity/Relationship Graphs: Unifying Data Modelling and Taking Data Integrity Management to the Next Level. In *Proceedings of the 2025 International Conference on Management of Data (SIGMOD '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 18 pages.

SIGMOD '25, June 22–27, 2025, Berlin, Germany

1 INTRODUCTION

This work bring together classical Entity/Relationship (E/R) modeling [7] and modern graph databases in the form of property graphs, for their mutual benefit and that of their communities.

On the one hand, graph data management may arguably constitute the most exciting development in data management for decades [18]. Indeed, many different graph data models emerge in response to the demand from applications. The property graph model [6] is emerging as a strong candidate for a future standard, and proposals for property graph schemata (PG-Schema [1]) and keys (PG-Keys [2]) have been developed carefully by academics and practitioners together. However, important and interesting questions still are what well-designed graph databases should be, and, in particular, how entity and referential integrity can be managed effectively and efficiently.

On the other hand, Chen's E/R model [7] constitutes a best breed of conceptual data models. The model captures entities and their relationships in a simple and easy-to-understand framework powerful enough to derive a formal data model. E/R models represent natural language features and complex sentence hierarchies graphically to serve as a basis for communicating how well the current model captures requirements for the target database [8]. Indeed, the graphical depiction in form of an E/R diagram is invaluable for effective communication between stakeholders that have different expertise. E/R diagrams capture those relational database schemata that are in Inclusion Dependency Normal Form, which is understood as the blueprint for well-designed relational databases [15, 16]. Interestingly, E/R diagrams are graphs, but their instances (E/R instances) are hierarchically nested tuples which are not graphs. Moreover, E/R models are conceptual data models, which means that they are usually translated into logical models first, such as the relational model, before they are implemented. In fact, there is no actual database system that natively manages E/R models and their instances. So, an interesting question is whether a more natural semantics can be assigned to E/R instances that can be managed natively.

As the main contribution of our work we propose the concept of *E/R graphs*, which we will define formally later. Informally, E/R graphs are property graphs that are instances of E/R diagrams. This informal definition enables us to explain the far-reaching benefits of E/R graphs in simple words. Firstly, they provide the classical E/R model with a graph-based semantics. Since E/R graphs are property graphs, there are database systems that can manage them natively, such as Neo4j. Secondly, E/R diagrams themselves are property graphs, which means that E/R graphs and E/R diagrams are a great fit, and unify classical E/R modeling, logical data modeling and modern graph data modeling. Thirdly, as E/R models capture well-designed databases, E/R diagrams and their graphs constitute well-designed graph databases.

In addition to bringing together classical E/R with modern graph modeling, E/R graphs have also important consequences for data

integrity management. Translating E/R diagrams into relational database schemata that are in IDNF constitutes an outcome that has been perceived as optimal for decades [5, 16]. Nevertheless, an overarching conundrum is that key attributes of referenced tables must be repeated in referencing tables as foreign key attributes. This is the only available mechanism to manage referential integrity in relational databases, which is also profound to joining tables when querying the databases.

Another major benefit of E/R graphs is that entity and referential data integrity can be managed without introducing property redundancy. Arguably, this constitutes the holy grail of data integrity management since every fact would be stored once only, minimizing sources of potential data inconsistency, and maximizing the efficiency of managing referential data integrity. Indeed, the alternative to redundantly duplicating key properties of referenced nodes as foreign key properties on referencing nodes, is to simply use a directed edge from the referencing to the referenced node. We will call such directed edges E/R links. This is as simple and natural as it gets, and a core feature of using graphs for data management. In fact, this is not possible in the relational model of data. However, in using E/R links as references, we may also need to use them as part of a key for the referencing node. Hence, we refer to keys that combine local properties, namely those available on the node, with E/R links as *E/R keys*, while we refer to keys that only use local properties as *property keys*. The concept of an E/R key is not new at all, but is part of the standard semantics for E/R diagrams [20]. However, E/R keys are not available in any database system as they get lost in translation to logical data models. Interestingly, even native graph database systems do not offer capabilities for E/R keys, but current support is limited to property keys. As a consequence, the redundant duplication of properties is a necessity to managing entity integrity with property keys. As we will show, this limitation holds back advances in data integrity management. In fact, our paper promotes the future use of E/R keys, in particular, as edges are a central feature of graphs, and - as we reveal - constitute a mechanism to implement referential integrity at the instance level. Also interestingly, the recently proposed class of PG-keys can fully express E/R keys (but not vice versa). However, we show that the implication problem for PG-keys is undecidable, while that of E/R keys can be decided in linear time. Hence, E/R keys constitute a computationally-friendly fragment of PG-keys. For full insight into the use of keys and E/R links for data integrity management in property graphs, we propose a whole spectrum of handling entity and referential integrity in E/R graphs. This ranges from i) a purely relational semantics in which properties are redundantly duplicated to manage referential integrity by key/foreign key properties and entity integrity by property keys, to ii) a full graph semantics in which no properties are duplicated redundantly and referential integrity is managed by E/R links while entity integrity is managed by E/R keys, with iii) a mixed semantics where only key properties are duplicated as foreign key properties when they contribute to the key of the referencing node, such that entity integrity can still be managed exclusively by property keys with a minimal use of property redundancy while referential integrity can be managed by E/R links or key/foreign key properties. While this fully explains the new spectrum of available alternatives in handling data integrity for well-designed graph data, the mixed approach minimizes property redundancy while still offering full support for managing entity and referential integrity in a currently available database system.

We summarize the contributions of our work as follows.

(1) We introduce the concept of E/R graphs as property graphs that are instances of E/R diagrams, and demonstrate that it bridges classical E/R with modern graph modeling.
(2) We show that E/R graphs define homomorphisms on E/R diagrams, therefore providing a natural graph semantics for E/R models, available for immediate use in some native graph database systems.
(3) We demonstrate that E/R diagrams are property graphs themselves, showing that E/R diagrams and graphs together unify conceptual, logical and graph data modeling. Indeed, E/R diagrams form the core of PG-Schema that captures hierarchical property graphs that are well-designed.
(4) We establish a comprehensive range of techniques for handling entity and referential data integrity in E/R graphs, ranging from a full relational to a full graph semantics. In particular, the full graph semantics uses E/R keys and E/R links to handle entity and referential integrity without any need to introduce property redundancy on nodes. In addition, a mixed semantics minimizes property redundancy while still being able to exclusively rely on property keys for entity integrity management.
(5) We show that E/R keys form a computationally-friendly fragment of PG-keys. In fact, we show that the implication problem for PG-keys is undecidable while that of E/R keys is decidable in linear time. This also signifies the power of PG-keys for future proposals of other fragments for different applications.
(6) Using experiments with the TPC-H benchmark, we quantify how well our different techniques handle entity and referential integrity maintenance in E/R graphs. While the full graph semantics offers the best benefits, the mixed semantics maximizes benefits by currently available technology.

Our work advocates how two strong lines of database communities and their research can be brought together to advance best practices for data modeling and data integrity maintenance. PG-Schemata and PG-Keys constitute fundamental proposals for schema and key languages in graph data management. E/R diagrams and E/R keys constitute fragments of these proposals that are important for graph data modeling and integrity management.

Organization. Section 2 summarizes basic definitions of the property graph model and discusses related work, while Section 3 summarizes basic definitions of the Higher-order Entity-Relationship model. We bring these two strands together in Section 4, showing that E/R diagrams are property graphs, proposing the new concept of E/R graphs, showing how the latter unify conceptual, logical, and graph modeling, demonstrating the undecidability of PG-keys, and the linear-time decidability of E/R keys. Section 5 proposes a full spectrum of handling data integrity in E/R graphs. Experiments are discussed in Section 6, quantifying how well the different semantics support data integrity maintenance in E/R graphs. We conclude in Section 7 where we also comment on future work.

2 GRAPHS WITH PROPERTIES

We recall the basics of the property graph model, discuss recent proposals for schema and key languages of property graphs, and mention other related work on graph modeling.

2.1 Property Graphs

The *property graph model* [6] is based on the following disjoint sets: $\mathcal{O}$ for a set of objects, $\mathcal{L}$ for a finite set of labels, $\mathcal{K}$ for a set of properties, and $\mathcal{N}$ for a set of values.

A *property graph* is a quintuple $G = (V, E, \eta, \lambda, \nu)$ where $V \subseteq \mathcal{O}$ is a finite set of objects, called *vertices*, $E \subseteq \mathcal{O}$ is a finite set of objects, called *edges*, $\eta : E \to V \times V$ is a function assigning to each edge an ordered pair of vertices, $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$ is a function assigning to each object a finite set of labels, and $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$ is a partial function assigning values for properties to objects, such that the set of domain values where ν is defined is finite. If $\nu(o, A)$ is defined, we write $\nu(o, A) = \downarrow$ and $\uparrow$ otherwise. Some examples of property graphs will follow soon.

2.2 PG-Schema

The recent proposal of PG-Schema [1] is a community-based effort of academics and practitioners to help with standardization efforts for a graph query language, in particular schema support. PG-Schema aims at providing a simple yet powerful formalism for specifying property graph schemata, including flexible type definitions supporting multi-inheritance, and expressive constraints. The formal syntax and semantics of PG-Schema is aimed at meeting principled design requirements of contemporary property graph management.

In relationship to our work, it is important to note that the authors of PG-Schema "deliberately target minimal data modeling capabilities and as a reference point ... take the most basic variant of Entity-Relationship (ER) diagrams ... as the ultimate lower bound in the expressiveness of conceptual modelling languages". As mentioned in [1], ER diagrams usually require the specification of a key for each of their entity and relationship types, which is viewed as a minimal requirement for entity integrity and typically maps to a primary key in relational databases. The other reason is not to overload ER diagrams with multiple key definitions. This does not mean, however, that multiple key constraints or other constraints cannot be specified. Indeed, ER modeling has brought forward many formalisms for integrity constraints, handsomely summarized in [20, Chapter 4].

While [1] views E/R diagrams as a lower bound for PG schemata, we reveal the benefits of adopting E/R diagrams as models for hierarchical property graphs, particularly in terms of capturing well-designed property graphs and handling data integrity maintenance.

2.3 PG-Keys

Similarly to the proposal of PG-Schema there is also a recent proposal of PG-Keys [2], which constitutes a flexible and powerful framework for defining key constraints designed by the Linked Data Benchmark Council's Property Graph Schema Working Group with members from industry, academia, and the ISO Graph Query Language standards group. PG-Keys can combine basic restrictions that require keys to be exclusive, mandatory, and singleton. The keys are applicable to nodes, edges, and properties since each of them represent valid real-life entities. "PG-Keys aims to guide the evolution of the standardization efforts towards making systems more useful, powerful, and expressive".

We show that E/R keys form the data modeling core of PG-Keys that is computationally attractive. Indeed, we show how PG-Keys can express E/R keys, but while the implication problem of PG-Keys is undecidable, that of E/R keys is decidable in linear time. In particular, the aim of PG-Keys is really to offer powerful and expressive support to different applications, while the aim of E/R keys is to support data integrity management for E/R graphs. In particular, E/R keys reveal how current constraint support for keys should be extended to make the most of graph features in terms of maintaining referential integrity.

2.4 Other Related Work

In the context of relational databases, the value of acyclicity has been recognized and documented very early [5]. Interestingly, there are recent efforts to merge different graph-based approaches on the basis of acyclicity as well [13]. Relational database schemata in Inclusion Dependency Normal Form (IDNF) are perceived as well-designed [16], due to strong technical arguments. Schemata in IDNF are acyclic, key- and foreign-key based, meaning that all functional dependencies are keys, and all inclusion dependencies are foreign keys. This ensures entity and referential integrity, the absence of value redundancy and sources of potential inconsistency, and reduces update inefficiencies [15]. Translating E/R diagrams into relational database schemata ensures they will be in IDNF, and every schema in IDNF can be translated into an E/R diagram.

The interest and range in graph database models has been large [3], and the recent book [6] provides a good overview of the property graph model. Similarly, there is extensive work on graph constraints [6, 12] and early work on property graph normalization [19].

Finally, there have been a range of efforts to translate conceptual models, such as UML and E/R models into graph databases [9, 10, 17, 21]. Perhaps closest to our work is the approach in [4] that sets out a formal design framework for practical property graph schema languages that is inspired by important elements of E/R models. However, they aim at proposing a formalism for general property graphs, not limited to acyclicity, and map entities to nodes, relationships to edges, and records to property-value pairs. In particular, mapping relationships to edges is not well founded. Firstly, forcing data modelers to model non-binary relationship as binary ones is not best practice, and also only possible at the cost of introducing expensive join constraints that preserve the equivalence of relations [20]. Secondly, mapping relationships to edges also means that no higher-order relationships can be modeled, which is again violating principles of data modeling [5, 14, 20], and standard relational database schemata with chains of foreign keys beyond length one could not be modeled (including, for example, TPC-H). Indeed, edges could then not be used as the target of other edges, without using a more complicated graph model. In contrast, we will directly use the most natural conceptual data model. In fact, we propose to use Higher-order Entity-Relationship Diagrams as models for hierarchical property graphs [20]. These always ensure

acyclicity and hierarchically structure objects of arbitrary arity and depth [20]. While we also map entities to nodes, and (nested) records to property-value pairs, we map relationships to nodes as well, reserving edges (E/R links) for the use of referential integrity. Note that this follows best data modeling practice [5, 14, 20] and has also been followed by XML []. The crucial observation of our paper is that E/R links provide a fundamental mechanism for ensuring referential integrity without redundant use of properties/attributes. Viewing E/R diagrams directly as models of property graphs has the benefit that the entire E/R framework becomes available for the modeling of hierarchical property graphs, that is, E/R graphs, including the rich literature on E/R constraints. The latter, however, is beyond the focus of the current paper. In addition, our approach also establishes a property graph semantics for the Higher-Order Entity-Relationship model, with all the advantages concerning data modeling and integrity management revealed in this paper.

In summary, while the literature on graph modeling is broad and deep, no previous work has used E/R diagrams directly as property graphs models, proposed the use of hierarchical property graphs as a graph-based semantics for E/R models, nor investigated how to manage entity and referential integrity by property graph models.

3 FUNDAMENTALS OF E/R MODELING

We provide a detailed summary of the syntax and semantics for the Higher-Order Entity/Relationship model [20]. Given this summary, it will become apparent how E/R diagrams can be directly viewed as models for hierarchical property graphs. In particular, it is of fundamental importance to utilize higher-order relationship types as these do not unnecessarily restrict the length of hierarchies between object types (that is, the length of key/foreign key chains in the relational model of data, or the length of directed paths in E/R graphs). This is important as requirements are typically provided in natural language, and higher-order relationship types can represent data stories about entities and their relationships that can be of arbitrary depth.

3.1 Entity types, Entities and Entity Sets

Entities form the atomic units of a data model. In the relational model, entities are records of tables that do not reference entities in other tables. In property graphs, entities are leaves. We can model entities of the same type by the same entity type, that is, entity types consist of a finite set of attributes (properties). In addition, any entity of any type needs to be uniquely identifiable by a subset of these attributes, which we call the key of this entity type.

Formally, an entity type E consists of a finite, non-empty set $attr(E)$ of attributes and a non-empty subset $id(E) \subseteq attr(E)$, which we call the *key* of E and every element $A \in id(E)$ is called a key attribute of E. Every attribute A has a domain $dom(A)$, which is a non-empty set of possible values we can assign for any entity on that attribute.

An entity e of entity type E assigns a value $e(A) \in dom(A)$ to every attribute $A \in attr(E)$. That is, $e : attr(E) \to \bigcup_{A \in attr(E)} dom(A)$ such that for all $A \in attr(E)$, $e(A) \in dom(A)$.

We denote the collection of entities over entity type E by $ent(E)$, which is the Cartesian product over the domains associated with the attributes of E. Formally, $ent(E) = \prod_{A \in attr(E)} dom(A)$.

An *entity set* of entity type E is a finite set $E^t \subseteq ent(E)$ where different entities have different values on some key attribute. That is, for all $e, e' \in E^t$ where $e \neq e'$, $e \mid_{id(E)} \neq e' \mid_{id(E)}$, in other words, there is some $A \in id(E)$ such that $e(A) \neq e'(A)$. Hence, values on key attributes enable us to uniquely identify every entity.

In the TPC-H benchmark schema, we have the following two entity types:

- Region=({regionkey, name, comment}, {regionkey}), and
- Part=({partkey,name, mfgr, brand, type, size, container, retailprice, comment},{partkey}).

3.2 Relationship types, Relationships, and Relationship Sets

Relationships constitute (complex) aggregations between (less complex) relationships and entities. It is of utmost importance that no cycles are introduced between relationships, as cyclic relationships would constitute self-references that are ill-defined. In other words, we maintain a strict hierarchy between relationships in order to establish a well-defined semantics where more complex relationships can be defined on top of less complex relationships. Entities constitute relationships that do not depend on any other entities or relationships.

For every positive integer k, a relationship type R of order k consists of

- a finite set $comp(R)$ of relationship types, called *components* of R, such that
  - if $k = 0$, then $comp(R) = \emptyset$,
  - if $k > 0$, then every $R' \in comp(R)$ is of order smaller than k and there is some $R' \in comp(R)$ of order $k - 1$.
- a finite set $attr(R)$ of attributes such that every $A \in attr(R)$ has a domain $dom(A)$, and
- a non-empty subset $id(R) \subseteq comp(R) \cup attr(R)$, called the *key* of R such that the components in $id(R) \cap comp(R)$ are called *key components* and the attributes in $id(R) \cap attr(R)$ are called *key attributes* of R.

Entity types may be understood as relationship types of order 0 which have a non-empty set of attributes but no component. Note that relationship types of order $k > 0$ do not need to have any attribute but they will have some relationship type of order $k - 1$ as a component.

If the order k of relationship type R is 0, then the relationships of R are the entities of R, that is, $rel(R) := ent(R)$. Otherwise, the order $k > 0$ of R is a positive integer, and the set $rel(R)$ comprises all relationships r of order k where

- every $R' \in comp(R)$ is mapped to a relationship $r(R') \in rel(R')$ of the order of R', and
- every $A \in attr(R)$ is mapped to a domain value $r(A) \in dom(A)$.

In summary, we may also write

$$rel(R) = \prod_{R' \in comp(R)} rel(R') \times \prod_{A \in attr(R)} dom(A).$$

Note that the collections of relationships over a relationship type R of order k are well defined, as all components of R have an order strictly less than k and relationships of order 0 are entities.

Just like entity sets, relationship sets must also satisfy the unique key property. More formally, a *relationship set* over relationship type R is a finite set $R^t \subseteq rel(R)$ such that for all $r, r' \in R^t$ such that $r \neq r'$, $r\,|_{id(R)} \neq r'\,|_{id(R)}$.

There are situations in which relationships involve multiple components of the same type, such as *home* : Team and *away* : Team to model two teams that play each other in a game where one team assumes the role of a home team and the other that of an away team. In addition, we may want to simply label a component to highlight its role. For that reason, we extend the definitions by permitting role-based components in the form of $l : R \in comp(R)$ with some label l.

Key attributes motivate the use of a foreign key semantics, in which for relationships r of order $k > 0$, we only use $rel(R) := \prod_{R' \in id(R) \cap comp(R)} rel(R') \times \prod_{A \in id(R) \cap attr(R)} dom(R)$ to define relationships and relationship sets of order k. Indeed, this semantics minimizes the redundant duplication of attributes from lower-order relationship types within higher-order relationship types.

Collectively, we may use the term *object type* as reference to relationship or entity types without specifying which. Similarly, we may use the term *object* (*object set*) as reference to relationship (*relationship set*) or entities (*entity set*) without specifying which. The order of an object type reflects the level of aggregation used to form any object of this type.

For the TPC-H benchmark, we have the follow relationship types and their orders. We ignore domains of attributes, and do not list all attributes for Lineitem (they are all shown in Fig. 1).

- Nation=({Region}, {nationkey, name, comment}, {nationkey}) of order 1,
- Supplier=({Nation}, {suppkey, name, address, phone, acctbal, comment}, {suppkey}) of order 2,
- Customer=({Nation}, {custkey, name, address, phone, mktsegment, acctbal, comment}, {custkey}) of order 2,
- PartSupp=({Part, Supplier}, {availqty, supplycost, comment}, {Part, Supplier}) of order 3,
- Orders=({Customer}, {orderkey, orderstatus, totalprice, orderdate, orderpriority, shippriority, clerk, comment}, {orderkey}) of order 3,
- LineItem=({PartSupp, Orders}, {linenumber, quantity, ..., shipmode, comment}, {Orders, linenumber}) of order 4.

Interestingly, we can see a mix of keys for these relationship types. Some contain only attributes, such as Nation, Supplier, Customer, and Orders, some contain only components, such as PartSupp, and some contain components and attributes, such as Lineitem. This mix of using components and attributes in keys is indicative of what happens in practice.

3.3 E/R Schemata and E/R Instances

Interestingly, the concept of referential integrity is deeply embodied in Entity/Relationship Schemata, as defined next.

An *Entity/Relationship Schema* (E/R Schema) is a non-empty, finite set $\mathcal{S}$ of object types such that for all relationship types $R \in \mathcal{S}$ and for all $R', l : R' \in comp(R)$, $R', l : R' \in \mathcal{S}$. So indeed, a higher-order type cannot exist without the lower-order types it depends on. In a relational database schema, a chain of components translates into a chain of foreign keys.

An E/R Schema $\mathcal{S}$ for the TPC-H benchmark schema contains the object types defined previously, namely Region, Part, Nation, Supplier, Customer, PartSupp, Orders, LineItem. Observe that this is indeed an E/R Schema. For example, for LineItem $\in \mathcal{S}$ and PartSupp $\in comp($LineItem$)$ we have indeed PartSupp $\in \mathcal{S}$.

These ideas on the abstract schema level transcend naturally to the instance level as follows.

An *Entity/Relationship instance* (E/R instance) $\mathcal{I}$ over E/R Schema $\mathcal{S}$ assigns to each entity type $E \in \mathcal{S}$, an entity set $\mathcal{I}(E)$ and to each relationship type $R \in \mathcal{S}$, a relationship set $\mathcal{I}(R)$ such that for each relationship $r \in \mathcal{I}(R)$ and for each object type $O \in comp(R)$, $r(O)$ is an object in $\mathcal{I}(O)$.

3.4 E/R Diagrams

Traditionally, conceptual schemata result from requirements engineering. They are used to communicate with various non-database experts that are stakeholders of the target database. Hence, conceptual data models must be easy to understand. The E/R model, in particular, enjoys intuitive visual representations in the form of E/R diagrams that facilitate such communication, are isomorphic to E/R Schemata, and represent natural language features. We repeat a standard definition of E/R diagrams [].

An *E/R Diagram* of E/R Schema $\mathcal{S}$ is a directed acyclic graph $\mathcal{G} = (V_G, E_G)$ such that the set V_G of vertices consists of the object types from the schema, $V_G := \mathcal{S}$, and the set E_G of directed edges, called *E/R links*, represents the component relationships between object types, $E_G := \{(R, R') \mid R' \in comp(R)\} \cup \{(R, R')$ with edge-label $l \mid l : R' \in comp(R)\}$. In particular, the order of object type $O \in \mathcal{S}$ is the maximum length across all directed paths from vertex $O \in V_G$ to any leaf in V_G.

An E/R diagram $\mathcal{G}$ is visualized as follows. An entity type $E = (attr(E), id(E)) \in V_G$ is drawn as a rectangle around the name E, attaching lines from the rectangle to the name A of each attribute $A \in attr(E)$, and underlining the names of key attributes $A \in id(E)$. Optionally, we may add the domains $dom(A)$ to every name A, that is, we write $A : dom(A)$. A relationship type $R = (comp(R), attr(R), id(R)) \in V_G$ is drawn as a diamond around the name R, attaching lines from the diamond to the name A of each attribute $A \in attr(R)$, and underlining the names of key attributes $A \in id(R) \cap attr(R)$. An E/R link $(R, R') \in E_G$ (with edge label l) is drawn as directed edge from the diamond around R to the diamond/rectangle around R' (with edge label l attached to the directed edge). In addition, if $R' \in id(R) \cap comp(R)$, we attach a dot to the directed edge to indicate that R' is a key component of R.

As a comprehensive example, Fig 1 shows a full E/R diagram of the TPC-H schema. In particular, the order of an object type is represented by different shades of gray, starting with white for entity types that are of order 0 up to the darkest gray for relationship type Lineitem of order 4.

4 E/R GRAPHS WITH PROPERTIES

In this section we will bring together the previous two sections. Indeed, we will define E/R diagrams as a special class of property graphs. Moreover, we will introduce E/R graphs as a special class of property graphs, namely property graphs that are isomorphic to E/R instances that conform to an E/R diagram. We show that

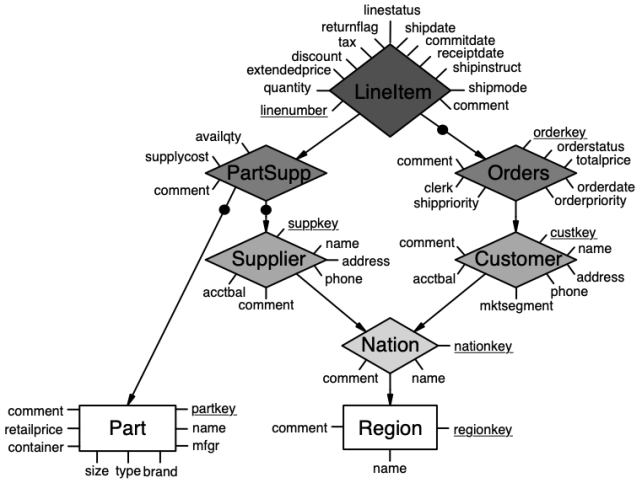


Figure 1: E/R Diagram for TPC-H

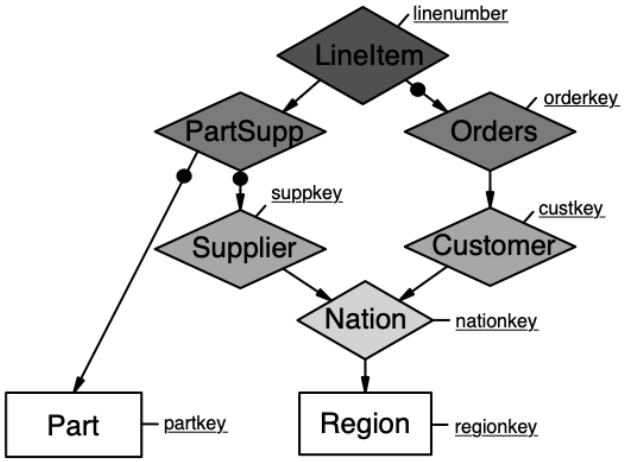


Figure 2: E/R Diagram for TPC-H with key attributes and components only

the implication problem of PG-keys is undecidable on property graphs that result from a simple translation of relational database schemata, while that of E/R keys is decidable in linear time over E/R diagrams. Finally, we illustrate different innovative approaches to managing referential integrity by E/R graphs.

4.1 Defining E/R Diagrams as Property Graphs

Essentially, E/R diagrams form a special class of property graphs, namely those that are non-attributed and acyclic. More formally, let $\mathcal{D} = (V, E)$ denote an E/R diagram.

The set $\mathcal{O}_D$ of object identifiers associated with $\mathcal{D}$ introduces a unique identifier i_o for each vertex in V (that is, each object type $O \in \mathcal{S}$), a unique identifier $i_{(o,o')}$ for each edge in E (that is, each component relationship $O' \in comp(O)$). That is, $\mathcal{O}_D = O_V \cup O_E$ where O_V consists of unique identifiers for each element of V, and O_E consists of unique identifiers for each object in E.

The set $\mathcal{L}_D$ of labels associated with $\mathcal{D}$ consists of the names O of object types in V, edge labels in $\mathcal{D}$ (resulting from labels l in component relationships $l : O' \in comp(O)$), and a distinguished label $\bullet$ used to represent dots on edges (resulting from key components $O' \in id(O)$).

The set $\mathcal{K}_D$ of properties associated with $\mathcal{D}$ consists of the attributes names in $attr(O)$ of all object types O in V.

The set $\mathcal{N}_D$ of values associated with $\mathcal{D}$ is the set all domains $dom(A)$ associated with any attribute A of any object type in $\mathcal{D}$. For flexibility and keeping values of non-key attributes optional, we distinguish sometimes between $dom(A)$ and $dom_\perp(A)$, where the latter contains the distinguished symbol $\perp$ for the null marker. As a principle of entity integrity and in line with primary keys we agree that whenever an attribute is part of a key, values are mandatory, that is, the attribute is `NOT NULL`.

Given over $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $\mathcal{D} = (V, E)$ is defined as follows:

(1) $V_D := O_V$,
(2) $E_D := O_E$,
(3) $\eta_D : E_D \to V_D \times V_D$ is defined by $\eta_G(o) = (i_o, i_{o'})$ where i_o is the unique identifier for node $O \in V$ and $i_{o'}$ is the unique identifier for node $O' \in V$ for every $(O, O') \in E$,
(4) $\lambda_D : V_D \cup E_D \to \mathcal{P}(\mathcal{L})$ is defined by $\lambda_D(i_o) = \{O\}$ for $O \in V$ with unique identifier $i_o \in O_V$, $l \in \lambda_D(i_{(o,o')})$ for $(O, O') \in E$ with edge label l and unique identifier $i_{(o,o')} \in O_E$, and $\bullet \in \lambda_D(i_{(o,o')})$ for $(O, O') \in E$ with key component $O' \in comp(O) \cap id(O)$ and unique identifier $i_{(o,o')} \in O_E$
(5) $\nu_D : (V_D \cup E_D) \times \mathcal{K}_D \to \mathcal{N}_D$ is defined by $\nu(i_o, A) = dom_\perp(A)$ for all attributes $A \in attr(O) - id(O)$ and all $O \in V$, and $\nu(i_o, A) = dom(A)$ for all key attributes $A \in attr(O) \cap id(O)$ and all $O \in V$.

Note that the definition also includes the key $id(O)$ of an object type O. The notation for key components $O' \in comp(O) \cap id(O)$ is maintained from E/R diagrams, and key attributes $A \in attr(O) \cap id(O)$ are identified as key properties by assigning the null-marker free domain $dom(A)$ to them. When visualizing properties in $\mathcal{G}_D$, we will still underline the key property $\underline{A}$, instead of writing down the property-value pair $A = dom(A)$, and simply write A for a non-key attribute A, instead of writing down the property-value pair $A = dom_\perp(A)$. Examples of these include $id(\textsc{LineItem}) = \{\textsc{Orders}, linenumber\}$, $id(\textsc{PartSupp}) = \{\textsc{Supplier}, \textsc{Part}\}$, and $id(\textsc{Orders}) = \{orderkey\}$. Note that our definition encodes the type definition (the domain constraints) of E/R diagrams as property-value pairs, using the domains of properties as values. Alternatively, we may also record the type definition in the form of a type definition for property graph schemata, as introduced in [4]. In our case, the resulting E/R graph model is a property graph itself, while the alternative would be a property graph schema in the sense of [4].

Our definition shows how every E/R diagram $\mathcal{D}$ gives rise to an E/R graph $\mathcal{G}_D$. Vice versa, it is also easy to see how to obtain the E/R diagram $\mathcal{D}$ from its E/R graph model $\mathcal{G}_D$. In other words, E/R schemata, E/R diagrams, and E/R graph models are isomorphic.

Typically, E/R diagrams just mention one (minimal) key for each object type, which corresponds to the focus on a PRIMARY KEY in relational databases. Similar to how relational databases permit the

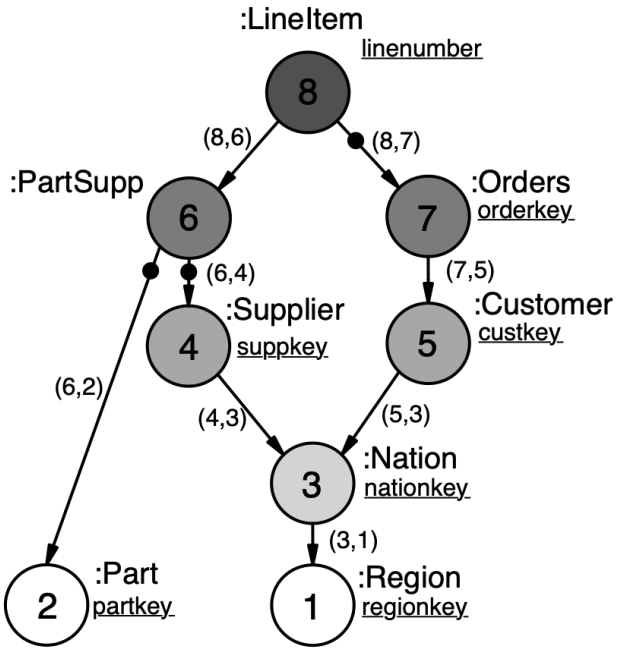


Figure 3: E/R Diagram for TPC-H as a property graph, where properties are limited to those resulting from key attributes

specification of candidate keys, we can also specify other keys on E/R diagrams. There are deep investigations on integrity contraints, even specifically on the E/R model [20], but this is beyond scope of the current article.

4.2 E/R graphs as Homomorphic Instances

We have just provided a technical definition of E/R diagrams as property graphs, called E/R graph models. In this section, we will provide a technical definition of E/R graphs that are instances of E/R graph models. In fact, E/R graphs are property graphs that map homomorphically to E/R graph models. Hence, E/R graphs are natural instances of E/R graph models.

Let $\mathcal{D} = (V, E)$ denote an E/R diagram for E/R schema $\mathcal{S}$. O_G is a set of object identifiers, $\mathcal{L}_G \subseteq \mathcal{L}_D - \{\bullet\}$ is a subset of labels, $\mathcal{K}_G \subseteq \mathcal{K}_D$ is a subset of features, and $\mathcal{N}_G \subseteq \bigcup_{A \in attr(O), O \in \mathcal{S}} dom(A)$ is a subset of values.

Let $\mathcal{G}_D$ denote an E/R graph model of E/R diagram $\mathcal{D}$. A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $O_G, \mathcal{L}_G, \mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ if and only if there is some function $h : O_G \to O_D$ that satisfies the following conditions:

(1) $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$
(2) for all $o \in E_G$, if $\eta_G(o) = (o_1, o_2)$, then $h(\eta_G) = (h(o_1), h(o_2)) = \eta_D(h(o))$ (edge-preserving)
(3) for all $o \in V_G$, $\lambda_G(o) = \lambda_D(h(o))$ (node label-preserving)
(4) for all $o \in E_G$, $\lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving)
(5) for all $o, o' \in E_G$, if $\eta_G(o) = (u, v), \eta_G(o') = (u, w) \in E_G$ and $\lambda_D(h(v)) = \lambda_D(h(w))$ and $\lambda_G(o) = \lambda_G(o')$, then $v = w$ (single-valued components, that is, equal target labels and role labels mean equal target nodes)
(6) for all $v \in V_G, p \in \mathcal{K}_G$, if $\nu_G(v, p) = val$, then $val \in \nu_D(h(v), p)$ (type-preserving)
(7) for all $v \in V_G, p \in \mathcal{K}_D$, if $\nu_D(h(v), p) = dom(p)$, then there is some $val \in dom(p)$ such that $\nu_G(v, p) = val$ (key properties must exist)
(8) for all $v \in V_G, w' \in V_D$, if $(h(v), w') \in E_D$, then there is some $w \in V_G$ such that $(v, w) \in E_G$ and $h(w) = w'$ (referential integrity)

Note characteristic (7): We do not have a domain-requirement whenever $\nu_D(h(v), p) = dom_\perp(p)$, as $\perp$ indicates that a value for this property does not need to exist.

We may define the order of a node $v \in V_G$ as the order of its homomorphic image $h(v) \in V_D$. Due to the homomorphism, any E/R graph is acyclic. As E/R diagrams embody referential integrity due to the inherent order of their vertices, E/R graphs - as instances of E/R graph models - inherit referential integrity for free (property 8).

For entity integrity, we may specify E/R keys for the property model graph $\mathcal{G}_D$ of an E/R diagram $\mathcal{D}$, namely as expressions $O(C, K)$ with $C = \{O_1, \ldots, O_n\} \subseteq comp(O)$ and $K = \{K_1, \ldots, K_m\} \subseteq attr(O)$ for any object type $O \in V_D$. An E/R graph G satisfies $O(C, K)$ if and only if for all $o, o' \in V_G$ such that $(o, o_i), (o', o_i) \in E_G$ for $i = 1, \ldots, n$, and $\downarrow = \nu_G(o, K_j) = \nu_G(o', K_j) = \downarrow$ for $j = 1, \ldots, m$, and $\lambda_G(o) = O = \lambda_G(o')$ and $\lambda_G(o_i) = O_i$ for $i = 1, \ldots, n$, then $o = o'$.

In fact, for G to be an E/R graph for $\mathcal{G}_D$ we require that the function h also satisfies the following condition:

(9) for all $O \in V_D$ such that $id(O) = C \cup K$ with $C = \{O_1, \ldots, O_n\}$ and $K = \{K_1, \ldots, K_m\}$, G satisfies the E/R key $O(C, K)$ (entity integrity)

In particular, the property of entity integrity is provided by specifying its condition in correspondence to object uniqueness in E/R instances. Note that we may specify multiple E/R keys on any object type, which every E/R graph then needs to satisfy. However, the principle of entity integrity requires at least one such E/R key, which is given by $id(O)$ in the specification of E/R schemata. This situation is similar to the use of primary keys in relational databases, which provide the primary index structure for accessing data physically. However, additional uniqueness constraints can be specified in the same way when more requirements for entity integrity are present. Note that E/R keys $O(C, K)$ do not stipulate object uniqueness whenever any value of any property in K is undefined. This is in line with the semantics of uniqueness constraints in SQL.

Fig. 4 illustrates the homomorphism from E/R graphs to E/R graph models on our running example of TPC-H, in particular the principle of entity integrity provided by the semantics of E/R keys. In Fig. 4, we illustrate the semantics of the E/R key :LineItem(: Orders, *linenumber*). Indeed, the two vertices o and o' have values matching on the property *linenumber*, and each has an E/R link to a node labeled Orders. If o and o' were different vertices, then the E/R graph would violate the E/R key. Hence, we conclude that $o = o'$.

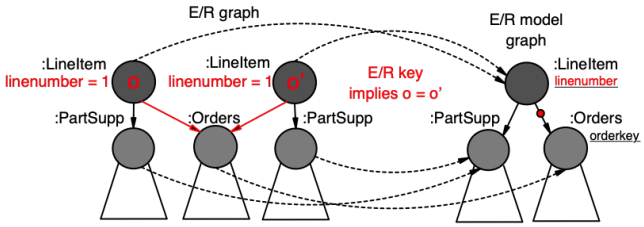


Figure 4: Illustration of E/R key on TPC-H Example

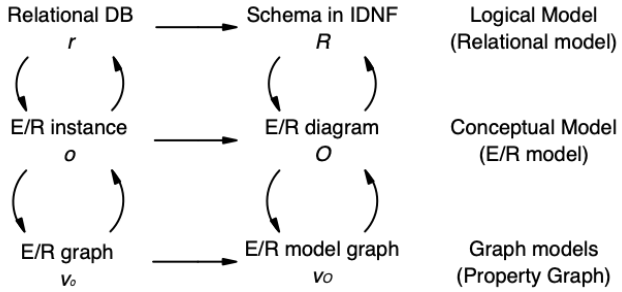


Figure 5: Mappings between instances and models that represent best data modeling practice

4.3 Graphical Core Unifying Data Modeling

Bringing together the previous sections, we have recalled the definitions of E/R instances and E/R diagrams, identified the latter as a special class of property graphs, and introduced E/R graphs as homomorphic instances of E/R graph models. Indeed, E/R instances are isomorphic to E/R graphs as the strict orders of objects and vertices enables us to easily translate one into the other.

Interestingly, E/R instances have really been defined as instances of E/R schemata. Previous work has not had defined a graph semantics, which is a contribution of the current paper by defining E/R graphs. Moreover, E/R instances have not directly been implemented in any logical data model. Best practice translates E/R diagrams and E/R instances into relational database schemata and relational databases, respectively. However, the graph semantics of E/R diagrams gets lost in these translations. More specifically, E/R links are replaced by key/foreign key relationships in relational database schemata. This is achieved by introducing attribute redundancy through the repetition of key attributes from referenced tables as foreign key attributes on referencing tables.

E/R modeling constitutes best modeling practice: features of E/R models can represent the structures inherent in natural languages, so are well suited to express requirements [8]; E/R diagrams serve as communication tool between database and domain experts [7]; E/R models result in relational schemata that are in Inclusion Dependency Normal Form [15, 16], which is ideal for entity and referential integrity maintenance in the context of relational databases, provides a logical grouping for database querying, and enjoys nice computational properties [15]. Vice versa, schemata in IDNF can be translated back into E/R diagrams, and similarly can their instances be translated into one another [16].

In the current work, we have introduced E/R graphs as the new kid on the block. They provide a graph semantics to E/R diagrams, constitute homomorphic instances of E/R graph models, and are property graphs, all in one. Specifically, they natively utilize edge relationships instead of introducing attribute redundancy. This does not only unify conceptual, logical and graph data modeling, but has also a profound impact on integrity management, as we will see soon.

Fig. 5 illustrates the situation. Relational database schemata in IDNF, E/R diagrams, and E/R graph models can all be translated into one another, and so can their instances. Together they form the core of best data modeling practice. On the one hand, relational database schemata in IDNF use keys, that are locally defined on their relation schemata, to enforce entity integrity, and foreign keys across relation schemata to enforce referential integrity. The latter, in particular, requires attribute redundancy, causing bottlenecks in referential integrity maintenance. On the other hand, E/R graph models use E/R keys to enforce entity integrity, and make direct use of E/R links to manage referential integrity. Indeed, E/R graphs appear to offer the first structure that only needs to store every fact once. Even E/R instances comprise attribute redundancy as the projection of higher order relationships to any component repeats at least the key attributes of the component. This already constitutes a loss of the "edginess" inherent in E/R diagrams.

It is relatively easy to see how E/R instances are mapped into E/R graphs, and vice versa. Indeed, the mapping can be defined recursively using the order of objects and vertices. Entities e with value $e(A) = val$ on attribute A can be mapped to leaf vertices v_e with $\nu(v_e, A) = val$, and relationships r of order $k + 1$ with relationship $r' = r(C)$ of order less than $k + 1$ on component C and value $r'(A) = val$ on attribute A are mapped to vertices v_r of order $k + 1$ with direct edges $(v_r, v_{r'})$ and $\nu(v_r, A) = val$. For object sets, key attributes and components ensure uniqueness, and this translates directly to E/R graphs, as seen before. The translation works similarly from E/R graphs to E/R instances.

In what follows, we will investigate the class of E/R keys on E/R graphs. Indeed, we will show that reasoning about PG-keys, as recently introduced, is infeasible. In fact, their implication problem is already undecidable for its relational fragment that does not even use edges in property graphs. In contast, the implication problem of E/R keys over E/R graphs can be decided in linear time. Of course, E/R keys are less expressive than PG-keys, but E/R keys target the handling of entity and referential integrity on databases that originate from best data modeling practice. We will see later on that the use of E/R graphs and keys avoid attribute redundancy in databases, therefore leading to superior performance in integrity maintenance, in particular as referential integrity comes for free.

4.4 Undecidability of PG-Keys

Firstly, we establish the undecidability for the implication problem of PG-keys over property graphs that originate from simple relational translations. Hence, the expressiveness of PG-keys has its price as reasoning about them is infeasible already on property graphs without edges. The undecidability is a direct consequence of the PG-keys' ability to express keys and foreign keys over relational instances, and the fact that the implication problem for keys by keys and foreign keys is undecidable [11].

First, we define a *relational translation* of relational databases into property graphs. Let $\mathcal{S}$ denote a relational database schema together with a finite set $\Sigma_{\mathcal{S}}$ of keys and foreign keys. In fact, $(\mathcal{S}, \Sigma_{\mathcal{S}})$ is translated to a property graph $\mathcal{G}_{\mathcal{S}}$ with a set $\Sigma_{\mathcal{G}_{\mathcal{S}}} = \{\sigma_{\mathcal{G}_{\mathcal{S}}} \mid \sigma \in \Sigma_{\mathcal{S}}\}$ of PG-keys by mapping

- every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $: R$,
- every attribute $A \in R$ to a property A on the node v_R,
- every key $\sigma = \{A_1, \ldots, A_n\}$ over R to a PG-key $\sigma_{\mathcal{G}_{\mathcal{S}}}$ over v_R:
  For $x : R$ IDENTIFIER $x.A_1, \ldots, x.A_n$
- every foreign key $\sigma = R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ to a PG-key $\sigma_{\mathcal{G}_{\mathcal{S}}}$ over v_R:
  For $x : R$ IDENTIFIER $x.A_1, \ldots, x.A_n, y$ WITHIN $(x), (y : S)$
  WHERE $x.A_1 = y.B_1, \ldots, x.A_n = y.B_n$.

In addition, relational database instances $\mathcal{I}(\mathcal{S})$ over $\mathcal{S}$ are translated into instances $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ over $\mathcal{G}_{\mathcal{S}}$ by mapping every tuple t of every relation $\mathcal{I}(R) \in \mathcal{I}(\mathcal{S})$ to a vertex $v_t \in V_{\mathcal{I}(\mathcal{G}_{\mathcal{S}})}$ with label $: R$ and properties $\nu(v_t, A) := t(A)$. In plain words, records are simply transformed into vertices with properties.

The *implication problem for relational PG-keys* is to decide whether for every given $(\mathcal{S}, \Sigma_{\mathcal{S}} \cup \{\varphi\})$, every instance $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ over $\mathcal{G}_{\mathcal{S}}$ that satisfies all elements of $\Sigma_{\mathcal{G}_{\mathcal{S}}}$ will also satisfy $\varphi_{\mathcal{G}_{\mathcal{S}}}$.

Theorem 4.1. *Under relational translations, the implication problem of relational PG-keys is undecidable.*

Proof. We reduce the implication problem of PG-keys over relational translations to that of keys by keys and foreign keys over relational databases, which is undecidable [11].

Let $\mathcal{I}(\mathcal{S})$ be an instance of $\mathcal{S}$ that satisfies the set $\Sigma_{\mathcal{S}}$ of keys and foreign keys but violates the key φ. By the relational translation, the corresponding instance $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ over $\mathcal{G}_{\mathcal{S}}$ will satisfy the set $\Sigma_{\mathcal{G}_{\mathcal{S}}}$ of relational PG-keys but violate the relational PG-key $\varphi_{\mathcal{G}_{\mathcal{S}}}$.

Vice versa, let $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ be an instance of $\mathcal{G}_{\mathcal{S}}$ that satisfies the set $\Sigma_{\mathcal{G}_{\mathcal{S}}}$ of PG-keys but violates the PG-key $\varphi_{\mathcal{G}_{\mathcal{S}}}$. By the relational translation, the corresponding instance $\mathcal{I}(\mathcal{S})$ over $\mathcal{S}$ will satisfy the set $\Sigma_{\mathcal{S}}$ of keys and foreign keys but violate the key φ. □

4.5 E/R keys to Manage Entity Integrity

Previously, we have already defined the syntax and semantics for E/R keys $O(C, K)$. Indeed, E/R keys form a special class of PG-keys. In fact, it is easy to see that an E/R key $O(\{O_1, \ldots, O_n\}, \{K_1, \ldots, K_m\})$ for $\mathcal{G}_D$ is satisfied by an E/R graph G for $\mathcal{G}_D$ if and only if G satisfies the following PG-key

$$\text{FOR } (x : O) \text{ IDENTIFIER } x.K_1, \ldots, x.K_m, y_1, \ldots, y_n \text{ WITHIN } (x) \to (y_1 : O_1), \ldots, (x) \to (y_n : O_n).$$

The *implication problem for E/R keys* is to decide whether for every E/R graph model $\mathcal{G}_D$ and every set $\Sigma \cup \{\varphi\}$ of E/R keys for $\mathcal{G}_D$, every E/R graph that satisfies all E/R keys in Σ also satisfies φ.

Theorem 4.2. *The implication problem of E/R keys over E/R graph models can be finitely axiomatized by the following rule*

$$\frac{O(C, K)}{O(CC', KK')}$$

which we call the extension rule.

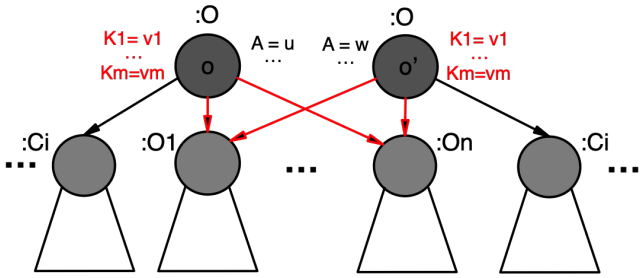


Figure 6: E/R graph in Completeness Proof of Theorem 4.2

Proof. We need to show the soundness and completeness of the extension rule for the implication of E/R keys over E/R graph models. Soundness means every E/R key that can be inferred from a given set Σ of E/R keys is also implied by Σ. This is easy to see as every E/R graph that violates $O(CC', KK')$ also violates $O(C, K)$.

It remains to demonstrate completeness, that is, we need to show that every E/R key that is implied by a given set Σ of E/R keys can also be inferred using applications of the extension rule only. We will show the contrapositive, and therefore assume that we cannot infer $O(C, K)$ from Σ using the extension rule. We need to show that there is some E/R graph G for the E/R graph model that satisfies Σ but violates $O(C, K)$. We construct an E/R graph from two distinct copies rooted at the node with label $: O$ in the E/R graph model, including distinct values on all properties of all nodes. For each $O_i \in C$, we merge the two nodes labeled by O_i, and for all $K_j \in K$, we use the smaller of the two values assigned to the root nodes o and o' on this property. Hence, any two distinct nodes have matching values on a property if and only if that property is part of K. The E/R graph is illustrated in Fig. 6. By construction, the resulting E/R graph violates $O(C, K)$. Assume there is some E/R key $O'(C', K') \in \Sigma$ that is violated by the E/R graph. Hence, there need to be two distinct vertices with the label O' that have, for each $O_i \in C'$, directed edges to the same vertex labeled by O_i, and matching values on all properties in K'. However, by construction, if two distinct vertices in the E/R graph have matching values on any property K_0, then $K_0 \in K$ and the two vertices must be the two roots. Also, whenever directed edges from two different vertices meet in the same vertex with label O_0, then $O_0 \in C$ and the two vertices must be the roots. Hence, if the E/R graph violates the standard E/R key $O'(C', K')$, then $C' \subseteq C$, $K' \subseteq K$ and $O' = O$. In other words, $O(C, K)$ can be inferred from Σ by a single application of the extension rule, in contradiction to our assumption. Consequently, the E/R graph satisfies Σ and violates $O(C, K)$, which proves completeness. □

There are various interesting consequence of Theorem 4.2. Firstly, we can state a very simple and efficient algorithm for deciding implication of E/R keys.

For a given set $\Sigma \cup \{O(C, K)\}$ of E/R keys over a given E/R graph model, it holds that Σ implies $O(C, K)$ if and only if there is some E/R key $O(C', K')$ in Σ such that $C' \subseteq C$ and $K' \subseteq K$. Hence, we obtain the following result as a corollary of Theorem 4.2.

Corollary 4.3. *The implication problem of E/R keys over E/R graph models is decidable in time linear in the input.* □

Hence, E/R keys form an expressive and computationally-friendly fragment of PG-keys. They are also very natural since they correspond to E/R keys over E/R schemata and diagrams, so are already known in best data modeling practice.

Another interesting consequence of Theorem 4.2 is that we can define the concept of a minimal E/R key. In fact, an E/R key $O(C, K)$ is minimal for a given set Σ of E/R keys if and only if Σ implies $O(C, K)$ and there is no E/R key $O(C', K')$ implied by Σ such that $C' \subseteq C$ and $K' \subset K$. Hence, the notion of minimal keys over relational databases extends naturally to that of minimal E/R keys for E/R graph models.

E/R keys are expressive as they handle entity integrity in the context of best data modeling practice. As their implication problem is decidable in linear time, while that of PG-keys is undecidable, E/R keys form a computationally-friendly fragment of PG-keys. Interestingly, current graph database systems can only specify uniqueness constraints with multiple properties, not including components. As we will see in later sections, this makes it necessary to redundantly repeat properties across nodes, effectively disallowing the use of edges in uniqueness constraints. Enabling the use of edges in keys would ensure full use of graph features and, as we demonstrate, advance integrity management to new levels.

4.6 New Approaches to Referential Integrity

A critical novelty of E/R graphs is their ability to handle referential integrity by the use of E/R links. Since E/R instances and relational database instances are not graph-based, edges are unavailable and referential integrity has been enforced by repeating attributes of lower-order object types in higher-order object types that contain them as components. In relational databases, for example, foreign key attributes reference the key attributes of parent tables, and updates of values on these attributes need to be propagated consistently across all redundant attributes in an effort to maintain referential integrity.

As an illustration we will discuss different implementations of referential integrity in our running example on TPC-H. The key *id*(LineItem) is LineItem(:Orders,*linenumber*).

Firstly, we may choose to map TPC-H relations to E/R graphs by repeating all attributes and introducing edges. In this case, we may express the E/R key as the PG-key:

For (x : LineItem) IDENTIFIER $x.linenumber, y$ WITHIN $(x) \rightarrow (y$: Orders) WHERE $x.orderkey = y.orderkey$

which makes use of both the E/R link from vertex :LineItem to vertex :Orders, and the equality of values on the property *orderkey* between both vertices. Indeed, the *orderkey* property is redundantly repeated on LineItem to provide a reference for the *orderkey* of :Orders. However, if we use this redundant repetition of properties, then we do not require the directed edge. Hence, we may also express the E/R key by:

For (x : LineItem) IDENTIFIER $x.linenumber, y$ WITHIN $(x), (y$: Orders) WHERE $x.orderkey = y.orderkey$

which corresponds to a strict semantics from relational databases where edges are not available. Alternatively, we may not make use of attribute redundancy, but can exclusively rely on edges. Such approach would follow the graph semantics of E/R graphs:

FOR (x : LineItem) IDENTIFIER $x.linenumber, y$ WITHIN $(x) \rightarrow (y$: Orders).

The latter perspective offers a completely new approach towards handling referential integrity, which does not require any redundant storage of the same fact across objects. This is not possible in relational databases or E/R instances.

In our experiments, we will investigate the profound implications of this approach on handling referential integrity at operational level.

5 DATA INTEGRITY IN E/R GRAPHS

We discuss the new opportunities E/R graphs offer for data integrity management. Firstly, we will demonstrate how the use of edges in E/R graphs eliminates the need for property redundancy to handle referential integrity, but requires E/R keys to manage entity integrity. We investigate three principled choices from a whole spectrum with complementary levels of using property redundancy and edges. Subsequently, we define mappings from (instances of) relational databases in IDNF to E/R graph (models) for each of the choices. These mappings will also serve as preparation for our experiments in the subsequent section.

5.1 Property Redundancy in E/R graphs

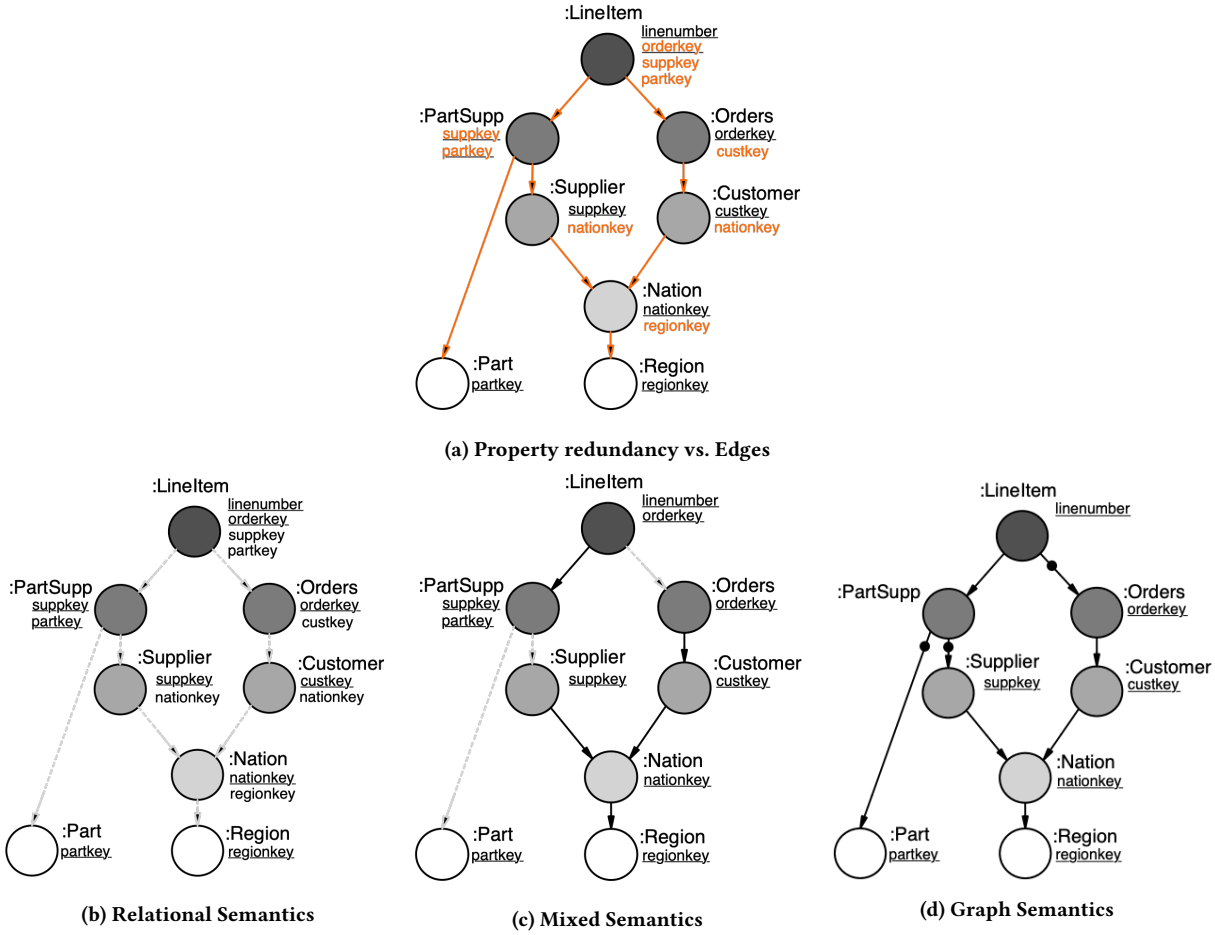
Our definition of E/R graphs as property graph-based instances of E/R diagrams introduces the opportunity to handle referential integrity without property redundancy. In fact, each component relationship $O' \in comp(O)$ introduces a key/foreign key relationship within a relational database. If $\{A_1, \ldots, A_n\}$ denotes the key on relation schema $R_{O'}$ that corresponds to object type O', then we have the foreign key $[A_1, \ldots, A_n] \subseteq R_{O'}[A_1, \ldots, A_n]$ on the relation schema R_O that corresponds to object type O. Here, the foreign key attributes $A_1, \ldots, A_n$ are redundant duplicates on R_O to handle referential integrity. However, in E/R graphs we simply use a directed edge from any vertex labeled : O to any vertex labeled : O', without repeating any key properties of O' on O.

Of course, we can also introduce property redundancy in E/R graph (models) by duplicating properties $P_{A_1}, \ldots, P_{A_n}$ of the key $\{A_1, \ldots, A_n\}$ over $R_{O'}$ on the relation schema R_O, and introduce the PG-key

For $(x : O)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}, y$ WITHIN $(x),(y : O')$
WHERE $x.P_{A_1} = y.P_{A_1}, \ldots, x.P_{A_n} = y.P_{A_n}$

to maintain referential integrity on E/R graphs. Whenever $O' \in comp(O) \cap id(O)$, we may also replace $O' \in C$ by the properties $P_{A_1}, \ldots, P_{A_n} \in K$ in $O(C, K)$. Doing this consistently, we can enforce entity integrity by E/R keys of the form $O(\emptyset, K)$. Hence, E/R graph models provide us with two alternatives for *every* component relationship: use property redundancy for referential integrity and property keys for entity integrity, or use E/R links for referential integrity and E/R keys for entity integrity. These alternatives are not exclusive or uniform. Note that we do not have that choice in any other data models considered before. In E/R graphs, we have that choice for every component relationship on every object type.

This choice is illustrated on the E/R graph model of TPC-H in Fig. 7a. Note that for each property highlighted in orange, there is an E/R links from the vertex with that property to the vertex on which that property contributes to the key. For each combination



(a) Property redundancy vs. Edges

(b) Relational Semantics

(c) Mixed Semantics

(d) Graph Semantics

Figure 7: E/R graph models for TPC-H under different semantics, only showing those attributes that are part of keys

of such a property and its corresponding E/R link, we have a choice between the property or the edge. If we choose the property but not the edge, we introduce property redundancy. Keeping the edge in addition to the property would introduce edge redundancy as well.

The remainder of Fig. 7 lists three principled choices, representing varying levels of property redundancy. In Fig. 7b we illustrate a pure relational semantics where we always choose property redundancy, representative for the case of relational databases. All orange properties from Fig. 7a are now black and all edges are grey and dotted, indicating that such an edge is redundant whenever we choose its properties. In Fig. 7c we only introduce redundancy on properties for which its corresponding component belongs to a key. In other words, we only introduce redundancy on those orange properties that are underlined, but no others. In this case, note how only three of the edges (those in grey and dotted) would be redundant. This is not an arbitrary choice, but to preserve the ability to handle entity integrity locally on each object type with property keys. That is, under relational and mixed semantics, we only require property keys (recall that these are E/R keys that only comprise attributes without components). In Fig. 7d, we do not introduce any property redundancy (none of the properties highlighted in orange in Fig. 7a), making full use of edges and their ability to handle referential integrity without introducing property redundancy. Note how all edges are required in this case, and how we need to make full use of E/R keys and the components they contain.

Table 1: Integrity Management Under Different Semantics

Semantics	Referential integrity	Entity integrity
Relational	Property redundancy	Property keys
Mixed	Mixed	Property keys
Graph	E/R links	E/R keys

Table 1 summarizes how the different semantics control entity and referential integrity in E/R graphs. On the one extreme, relational semantics uses exclusively property redundancy to manage both referential and entity integrity. Hence, no edges are required. On the other extreme, graph semantics uses exclusively E/R links to manage referential integrity, and therefore uses components as part of E/R keys whenever these belong to the keys. Mixed semantics combines the use of property redundancy and E/R links, while still

managing entity integrity exclusively by property keys. In fact, the introduction of redundancy is limited to properties that belong to a key component, while other components are referenced using E/R links. Hence, mixed semantics minimizes the introduction of property redundancy (maximizes the use of E/R links) while managing entity integrity exclusively by property keys.

In subsequent sections, we will define three mappings that transform relational database schemata that are in IDNF (and their instances) into E/R graph models (and their instances), one for each of the three semantics. Applying these mappings to TPC-H, we can then conduct experiments to reveal insights on how well entity and referential integrity maintenance is supported under each semantics.

5.2 Relational Semantics

For our mappings, we assume that the given relational database schema $(\mathcal{S}, \Sigma_{\mathcal{S}})$ is the result of converting an E/R schema together with a set of constraints. In particular, $\mathcal{S}$ is in Inclusion Dependency Normal Form with respect to the given set $\Sigma_{\mathcal{S}}$ of keys and foreign keys. Our mappings will also translate instances, so we are also given a relational database $\mathcal{I}(\mathcal{S})$ over $\mathcal{S}$ that satisfies $\Sigma_{\mathcal{S}}$. Note that the order of a relation schema $R \in \mathcal{S}$ is the length of a longest key/foreign key chain from R to another relation schema $S \in \mathcal{S}$ on which no foreign key is defined.

In our first mapping, called *relational mapping*, we do not use any edges in the output E/R graph (model). The order of vertices in both graphs is defined as order of the relation schema from which the vertex results.

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R$ with $dom(A)$ we have a property-value pair $P_A = dom(A)$ on v_R,
- for every key $\{A_1, \ldots, A_n\}$ of R we have a property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key:
  For $(x : R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$
- for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R we have a PG-key:
  For $(x : R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x),$(y : S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \ldots, x.P_{A_m} = y.P_{B_m}$.

Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping every

- tuple $t \in \mathcal{I}(R)$ for every $R \in \mathcal{S}$ to a node v_t with label R such that, for all $A \in R$, $\nu(v_t, P_A) := t(A)$.

As an example, the relation schema LineItem with

- attributes *suppkey*, *partkey*, *orderkey*, *linenumber*, …, *comment*
- key {*linenumber*,*orderkey*} and
- foreign keys
  - [*suppkey*, *partkey*] ⊆ PartSupp[*suppkey*, *partkey*] and
  - [*orderkey*] ⊆ Orders[*orderkey*]

would have a corresponding vertex v_{LineItem} with label LineItem and properties $P_{orderkey}$, …, $P_{comment}$, an ER-key LineItem($\emptyset$, {*linenumber*, *orderkey*}) implemented as

For $(x :$ LineItem) IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}$

and further PG-keys

- For $(x :$ LineItem) IDENTIFIER $x.P_{suppkey}, x.P_{partkey}, y$ WITHIN $(x), (y :$ PartSupp) WHERE $x.P_{suppkey} = y.P_{suppkey}, \ldots, x.P_{partkey} = y.P_{partkey}$, and
- For $(x :$ LineItem) IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}, y$ WITHIN $(x), (y :$ Orders) WHERE $x.P_{orderkey} = y.P_{orderkey}$.

5.3 Mixed Semantics

In our second mapping, we only introduce redundancy for properties that are part of the distinguished key, while using edges to eliminate redundancy for other properties. Hence, we minimize the use of property redundancy (equivalently, maximize the use of edges) to maintain the exclusive use of property keys.

For each relation schema $R \in \mathcal{S}$, let K_R denote the set of attributes in R that are part of some minimal key, and let F_R denote the foreign-key attributes that participate in some foreign key over R. Let $R_{\text{red}} := R - (F_R - K_R)$ consist of all the attributes from R without those that are foreign-key but not attributes in any key. That is, remove all those attributes from R that belong to some foreign key but do not belong to any key of R. Note that every foreign key uniquely identifies objects from the component it references, and that all attributes of this foreign key are required for that. Hence, for every key on R, either all attributes of the same foreign key belong to that key, or none of them does. In particular, we cannot have a foreign key where some of its attributes belong to a key and some others do not. In summary, for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R we have either $\{A_1, \ldots, A_m\} \subseteq R_{\text{red}}$ or $\{A_1, \ldots, A_m\} \cap R_{\text{red}} = \emptyset$.

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for every attribute $A \in R_{\text{red}}$ with domain $dom(A)$ we have a property-value pair $P_A = dom(A)$ of v_R,
- for every key $\{A_1, \ldots, A_n\}$ of R, we have property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key:
  For $(x : R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$
- for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $\{A_1, \ldots, A_m\} \subseteq R_{\text{red}}$, we have a PG-key:
  For $(x : R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x),$(y : S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \ldots, x.P_{A_m} = y.P_{B_m}$, and
- for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $\{A_1, \ldots, A_m\} \cap R_{\text{red}} = \emptyset$, we have
  - a directed edge (v_R, v_S)
  - a PG-key:
    For $(x : R)$ IDENTIFIER x, y WITHIN $(x) \rightarrow (y : S)$.

Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping every

- tuple $t \in \mathcal{I}(R)$ for every $R \in \mathcal{S}$ to a node v_t with label R such that, for all $A \in R_{\text{red}}$, $\nu(v_t, P_A) := t(A)$
- foreign key/key relationship where for $t \in \mathcal{I}(R)$ we have a unique $s \in \mathcal{I}(S)$, such that $t[A_1, \ldots, A_m] = s[B_1, \ldots, B_m]$, based on the foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ on R where $\{A_1, \ldots, A_m\} \cap R_{\text{red}} = \emptyset$, to a directed edge (v_t, v_s).

Continuing our example, the relation schema LineItem from before has a corresponding vertex v_{LineItem} with label LineItem

and properties $P_{orderkey}$, $P_{linenumber}$, …, $P_{comment}$ (note that properties $P_{suppkey}$ and $P_{partkey}$ are not required), a property key LineItem($\emptyset$, {*linenumber*, *orderkey*}) implemented as

For (x : LineItem) IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}$

and further PG-keys

- For (x : LineItem) IDENTIFIER x, y WITHIN (x) $\rightarrow$ (y : PartSupp), and
- For (x : LineItem) IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}, y$ WITHIN (x), (y : Orders) WHERE $x.P_{orderkey} = y.P_{orderkey}$.

Note that relation schema PartSupp has the key {*partkey*,*suppkey*}.

5.4 Graph Semantics

For our last mapping, we make full use of edges to express entity and referential integrity, making it possible to avoid property redundancy completely. For each relation schema $R \in \mathcal{S}$, let $R_g := R - F_R$ consist of all the attributes from R without those that are foreign-key attributes. That is, remove all those attributes from R that belong to some foreign key of R. Furthermore, let C_R denote the set of all relation schemata $S \in \mathcal{S}$ such that there is some foreign key $R[X] \subseteq S[Y] \in \Sigma_{\mathcal{S}}$. For each relation schema, there is a distinguished key (the primary key), dictating which components are labeled as key components in the resulting E/R graph model.

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R_g$ with domain $dom(A)$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for the distinguished key K of R in Σ, we have an E/R key $R(C_K, P_K)$ where $C_K := \{S \in C_R \mid K \cap S \neq \emptyset\} = \{S_1, \ldots, S_m\}$ and $P_K := \{P_A \mid A \in K\} - \{P_A \mid A \in S, S \in C_K\} = \{P_{A_1}, \ldots, P_{A_n}\}$, implemented as:
  - directed edges $(v_R, v_{S_1}), \ldots, (v_R, v_{S_m})$ with label • each,
  - For (x : R) IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}, y_1, \ldots, y_m$ WITHIN (x) $\rightarrow$ (y_1 : S_1), …, (x) $\rightarrow$ (y_m : S_m)
- for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $S \notin C_K$, we have
  - a directed edge (v_R, v_S), and
  - a PG-key: For (x : R) IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x) $\rightarrow$ (y : S).
- for every other key X of R, we have an E/R key: $R(C_X, P_X)$ where $C_X := \{S \in C_R \mid X \cap S \neq \emptyset\} = \{S_1, \ldots, S_m\}$ and $P_X := \{P_A \mid A \in X\} - \{P_A \mid A \in S, S \in C_X\} = \{P_{A_1}, \ldots, P_{A_n}\}$, implemented as:
  - For (x : R) IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}, y_1, \ldots, y_m$ WITHIN (x) $\rightarrow$ (y_1 : S_1), …, (x) $\rightarrow$ (y_m : S_m) (the directed edges $(v_R, v_{S_1}), \ldots, (v_R, v_{S_m})$ already exist)

Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping every

- tuple $t \in \mathcal{I}(R)$ for every $R \in \mathcal{S}$ to a nodes v_t with label R such that, for all $A \in R_g$, $\nu(v_t, P_A) := t(A)$
- foreign key/key relationship where for $t \in \mathcal{I}(R)$ we have a unique $s \in \mathcal{I}(S)$, such that $t[A_1, \ldots, A_m] = s[B_1, \ldots, B_m]$, based on the foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R, to a directed edge (v_t, v_s).

Table 2: Details on the graph translations of TPC-H

	sf=0.01		*sf=0.1*		*sf=1*	
Semantics	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
Relational	86,805	0	866,602	0	8,661,245	0
Mixed	86,805	76,800	866,602	766,597	8,661,245	7,661,240
Graph	86,805	152,975	866,602	1,527,169	8,661,245	15,262,455

Continuing our example, the relation schema LineItem from before has a corresponding vertex v_{LineItem} with label LineItem and properties $P_{linenumber}$, …, $P_{comment}$ (note that properties $P_{suppkey}$, $P_{partkey}$, $P_{orderkey}$ are not required), an ER-key LineItem({Orders}, {*linenumber*}) implemented as

For (x : LineItem) IDENTIFIER $x.P_{linenumber}, y$ WITHIN (x) $\rightarrow$ (y : Orders)

and further PG-key

- For (x : LineItem) IDENTIFIER x, y WITHIN (x) $\rightarrow$ (y : PartSupp).

Note that relation schema Orders has the key {*orderkey*}, and relation schema PartSupp has the key {*partkey*,*suppkey*}.

6 EXPERIMENTS

We will now analyze experimentally how well entity and referential integrity can be managed by the use of relational, mixed, and graph semantics. For this purpose, we will translate the TPC-H database into Neo4j using different scaling factors and semantics, and then measure the effort of maintaining entity and referential integrity.

Our most critical findings are: Graph semantics with E/R links and E/R keys manages entity and referential integrity most efficiently, and can have savings over the other semantics in orders of magnitude. The main disadvantage is that E/R keys do not enjoy native support in graph database systems yet, including Neo4j. For that reason, mixed semantics is most efficient for managing entity and referential integrity within the limits of currently available support for property keys.

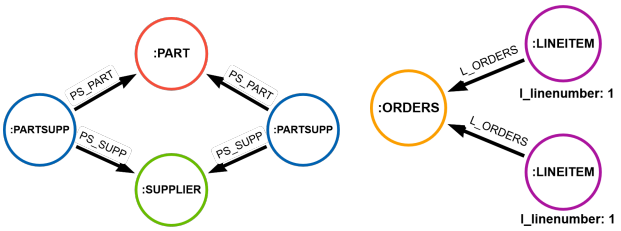
The TPC-H database schema is in Inclusion Dependency Normal Form, and diverse enough to quantify the advantages and disadvantages of the different semantics. This diversity refers to the availability of i) key/foreign key chains with different lengths, ii) E/R and property keys, iii) different data volumes.

6.1 Set up

For each of the three semantics, we translated the TPC-H database into Neo4j using scaling factors small (0.01), medium (0.1) and large (1). Table 2 summarizes how many nodes and edges are present in each translation under the different scaling factors.

For details of the experiments with results and artifacts, we refer the reader to our Github repository[1]. We used Neo4j and its query language Cypher in our experiments in conjunction with Python 3.9.13. These experiments were conducted on a 64-bit operating system with an Intel Core i7 Processor with 16GB RAM.

[1] https://github.com/graphdbexperiments/er_graph_experiments



(a) PS({Part, Supplier}, ∅) (b) L({Orders},{*linenumber*})

Figure 8: Patterns that violate E/R keys

6.2 Entity Integrity

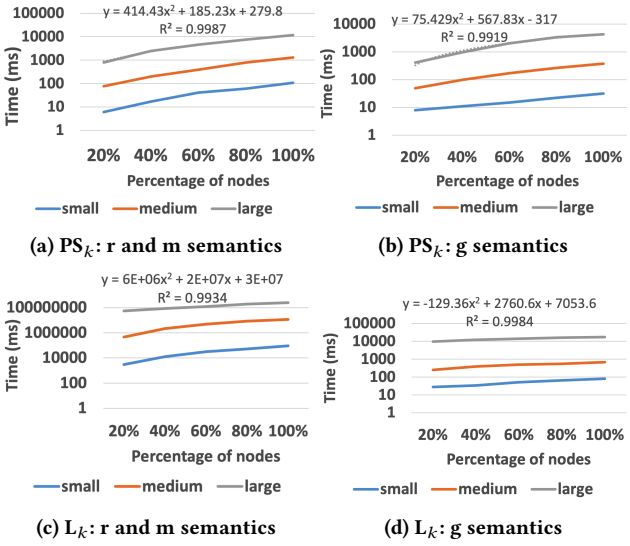
We will illustrate in this section how our graph semantics with E/R keys is more efficient for managing entity integrity than mixed and relational semantics with property keys. This is particularly interesting as property keys enjoy native support through unique indices within Neo4j, while E/R keys are not supported at all. The reason is that Neo4j and graph database systems pay particular attention to graph edges, which are E/R links within our E/R keys.

Our experiments have been conducted as follows. Based on our semantics, we have performed Cypher queries that validate the E/R key PS_k = PS({Part, Supp}, ∅) on PartSupp and the E/R key L_k = L({Orders}, {*linenumber*}) on Lineitem. Under relational (r) and mixed (m) semantics, we can specify these keys as unique constraints within Neo4j, and benefit from the resulting index. Under graph semantics (g), the corresponding E/R keys are not supported by Neo4j (or any other graph database system), so we cannot benefit from any index structure either. Under all semantics, we then validate entity integrity by running a corresponding Cypher query which returns all nodes that violate the key, so the key is satisfied if and only if the query answer is empty. Patterns that violate the E/R keys are illustrated in Fig. 8.

For each translation, Table 3 lists the E/R keys, their definitions as PG-keys, the Cypher queries that validate them, and the index structure that supports their validation. We used the following abbreviations: Part (P), Supplier, (S), PartSupp (PS), Orders (O) and Lineitem (L); and $K(s)$ for the E/R key K under semantics s, such as $\mathrm{PS}_k(r)$ for the E/R key PS_k under relational (r) semantics.

Fig. 9 illustrates how well E/R key validation scales under each of the relational/mixed and graph semantics, respectively. In our experiments, we averaged the time required for validating the E/R key PS({Part, Supplier}, ∅) for different percentages of nodes labeled PartSupp, namely 20, 40, 60, 80 and 100 percent, and likewise for the E/R key L({Orders},{*linenumber*}). The averages were taken over ten runs of the corresponding queries on each dataset resulting from the different scaling factors. It is visible that E/R key validation depends mostly on the underlying number of nodes considered, and these are also dependent on the three scaling factors considered, see Tab. 2.

Fig. 9 also shows that the time for validating E/R keys is much less under graph semantics than it is under relational/mixed semantics. Fig. 10 illustrates this observation more clearly as we fix the scaling factors and vary the semantics. For each of the three different scaling factors, and for each of the E/R keys it is very visible that validating entity integrity with E/R keys in graph semantics scales

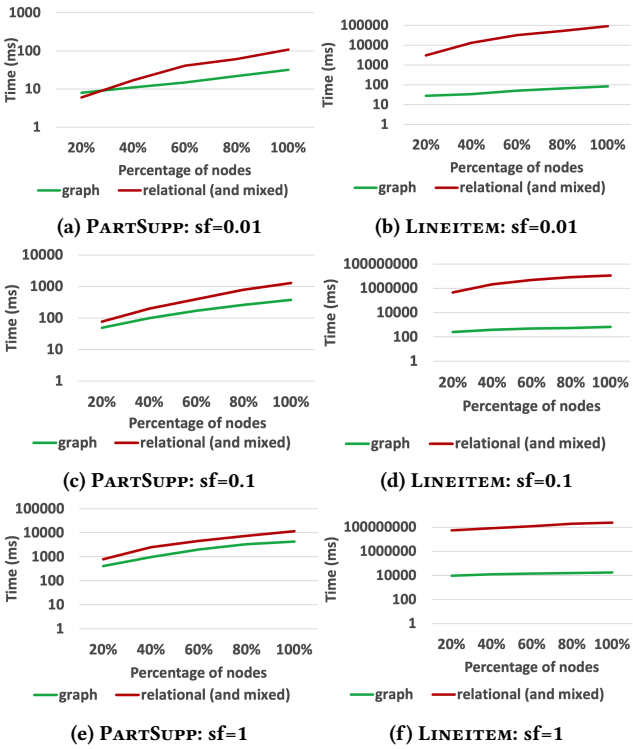


(a) PS_k: r and m semantics (b) PS_k: g semantics

(c) L_k: r and m semantics (d) L_k: g semantics

Figure 9: Scaling validation of E/R keys on TPC-H

much better than validating entity integrity with property keys in relational/mixed semantics.



(a) PartSupp: sf=0.01 (b) Lineitem: sf=0.01

(c) PartSupp: sf=0.1 (d) Lineitem: sf=0.1

(e) PartSupp: sf=1 (f) Lineitem: sf=1

Figure 10: TPC-H E/R key validation by different semantics

In Table 4 we have summarised the database hits (db hits) as well as the time (in ms) required to validate the E/R keys on PartSupp and Lineitem, listing all different scaling factors (sf) and semantics

Table 3: E/R keys used for validation experiments

Semantics	E/R key	PG-Key	Cypher Query	Index
Relational and Mixed	$PS_k(r) = PS_k(m)$: PS(∅,{partkey, suppkey})	FOR (ps:PS) IDENTIFIER ps.partkey, ps.suppkey	MATCH (ps1:PS), (ps2:PS) WHERE id(ps1) < id(ps2) AND ps1.partkey = ps2.partkey AND ps1.suppkey = ps2.suppkey RETURN ps1, ps2	PS(partkey, suppkey)
Graph	$PS_k(g)$: PS({P, S},∅)	FOR (ps:PS) IDENTIFIER p, s WITHIN (s:S) ← (ps) → (p:P)	MATCH q1 = (s:S)<-[]-(ps1:PS)-[]->(p:P), q2=(s:S)<-[]-(ps2:PS)-[]->(p:P) WHERE id(ps1) < id(ps2) RETURN q1, q2	no index
Relational and Mixed	$L_k(r) = L_k(m)$: L(∅,{orderkey, linenumber})	FOR (l:L) IDENTIFIER l.orderkey, l.linenumber	MATCH (l1:L), (l2:L) WHERE id(l1) < id(l2) AND l1.orderkey = l2.orderkey AND l1.linenumber = l2.linenumber RETURN l1, l2	L(orderkey, linenumber)
Graph	$L_k(g)$: L({O},linenumber)	FOR (l:L) IDENTIFIER l.linenumber, o WITHIN (l) → (o:O)	MATCH q = (l1:L)-[]->(o:O)<-[]-(l2:L) WHERE id(l1) < id(l2) AND l1.linenumber = l2.linenumber RETURN q	no index

(sem) considered. Interestingly, db hits measure the cost of running queries under index structures available in Neo4j, which is particularly relevant in our case as relational and mixed semantics are supported by unique indices resulting from the property key specifications. In contrast, there is no key and index support for E/R keys under graph semantics in Neo4j. Indeed, the presence of unique indices under relational/mixed semantics results in much more efficient estimates than the absence of indices under graph semantics. However, the actual validation time is much faster under graph semantics, testimony to the focus of graph database systems on exploiting edges. Indeed, E/R key validation is four orders of magnitude faster under graph semantics than relational/mixed semantics for the key on Lineitem and *sf=1* (18 seconds vs. 2.85 days), and that is without index support for graph semantics and available unique indices under relational/mixed semantics. This constitutes a great motivation for implementing E/R keys in graph database systems and make validation even faster.

Table 4: Measuring E/R key validation in TPC-H

E/R		*sf=0.01*		*sf=0.1*		*sf=1*	
key	sem	db hits	time (ms)	db hits	time (ms)	db hits	time (ms)
PS_k	r/m	16,002	108	160,002	1,288	1,600,002	11,669
	g	442,701	32	4,302,889	377	23,282,432	4,310
L_k	r/m	120,352	91,258	1,201,146	11,562,701	26,400,286	246,985,775
	g	842,452	82	8,860,929	686	88,530,831	17,696

6.3 Referential Integrity

A fundamental and exciting impact of graph databases is their ability to eliminate attribute redundancy, which results in tremendous savings when maintaining referential integrity. As this paper demonstrates, this is best achieved by our concepts of E/R graphs, E/R keys, and E/R links under the graph semantics we introduced. However, as E/R keys are not supported natively by graph database systems, we also introduced mixed semantics that maximizes savings by using property keys, so under the limitation of only using currently available technology. We use this section to illustrate and quantify our claims by experiments with translations of the TPC-H dataset into E/R graphs under relational, mixed and graph semantics.

In this context, we report on experiments that performed updates on properties. According to the E/R links of the TPC-H dataset that represent referential integrity constraints, such updates need to be propagated for maintaining referential integrity. For our experiments, we selected the following chains resulting from E/R links between the tables:

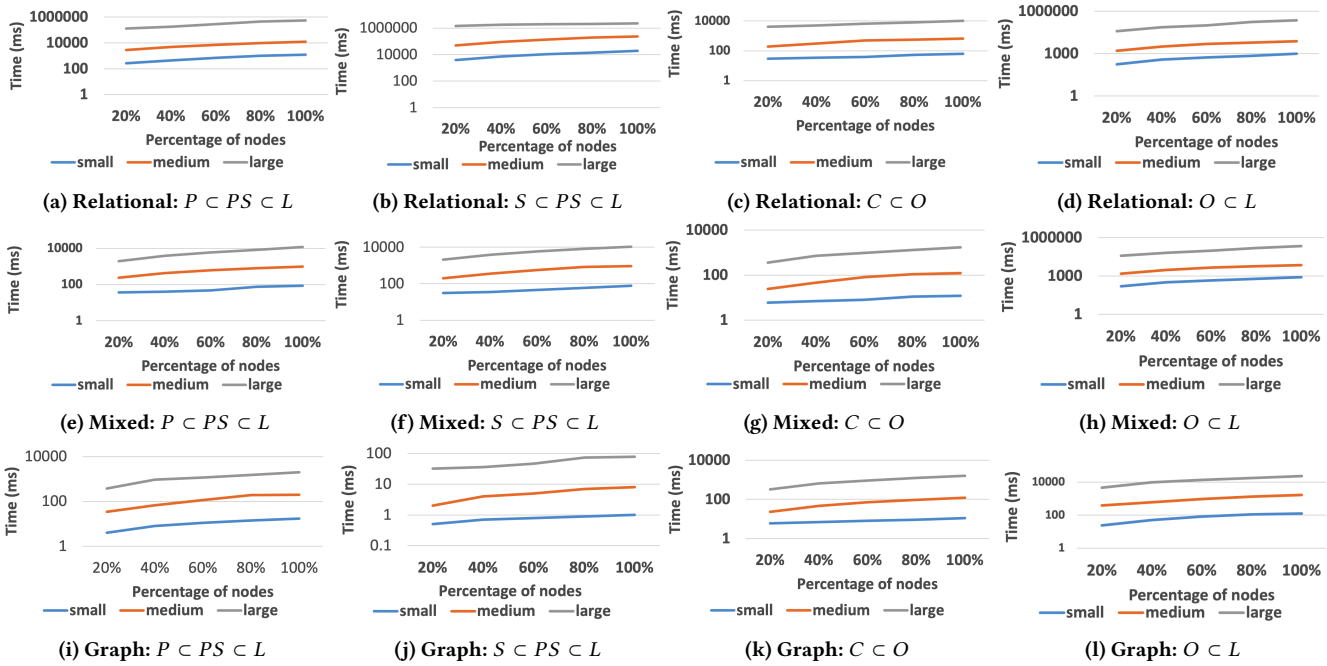
P_c: Part(P) ⊂ PartSupp(PS) ⊂ Lineitem(L)
S_c: Supplier(S) ⊂ PartSupp(PS) ⊂ Lineitem(L)
C_c: Customer(C) ⊂ Orders(O)
O_c: Orders(O) ⊂ Lineitem(L)

These chains are diverse in how they handle referential integrity across relational, mixed and graph semantics. As example, for the chain S_c : Supplier ⊂ PartSupp ⊂ Lineitem we have updated the property *suppkey* for different percentages of nodes labeled Supplier (for all semantics) which requires corresponding updates for property *suppkey* on PartSupp (for relational and mixed semantics) nodes and again for nodes labeled Lineitem (for relational semantics). In particular, as *suppkey* is a key attribute on PartSupp, it is duplicated on PartSupp nodes under relational and mixed semantics because they only use property keys, but not under graph semantics as it uses E/R keys. Likewise, *suppkey* is not a key attribute on Lineitem, so it is only duplicated on Lineitem nodes under relational but not under mixed or graph semantics. Table 5 shows the update operations in Cypher syntax we used in our experiments for the chain $S_c : S \subset PS \subset L$ under different semantics.

We conducted the same experiments for the other chains $P_c : P \subset PS \subset L$, $C_c : C \subset O$ and $O_c : O \subset L$. The chain $P_c : P \subset PS \subset L$ has very similar characteristics as the chain $S_c : S \subset PS \subset L$ with respect to the different semantics. For $C_c : C \subset O$ the property *custkey* is already not present on Orders nodes under mixed semantics, hence updates under mixed and graph semantics are the same. For $O_c : O \subset L$ the property *orderkey* is only present on Lineitem nodes under mixed semantics but not under graph semantics. Hence, updates under relational and mixed semantics remain the same in this case.

Table 5: Update propagation for $S \subset PS \subset L$ by semantics

Semantics	Update propagation
Relational	MATCH (s:S), (ps:PS), (l:L) WHERE ps.suppkey = s.suppkey AND l.suppkey = s.suppkey SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8), ps.suppkey = RIGHT(('00000000' + toString(ps.suppkey)), 8), l.suppkey = RIGHT(('00000000'+toString(l.suppkey)), 8)
Mixed	MATCH (s:S), (ps:PS) WHERE ps.suppkey = s.suppkey SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8), ps.suppkey = RIGHT(('00000000' + toString(ps.suppkey)), 8)
Graph	MATCH (s:S) SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8)



(a) **Relational:** $P \subset PS \subset L$ (b) **Relational:** $S \subset PS \subset L$ (c) **Relational:** $C \subset O$ (d) **Relational:** $O \subset L$

(e) **Mixed:** $P \subset PS \subset L$ (f) **Mixed:** $S \subset PS \subset L$ (g) **Mixed:** $C \subset O$ (h) **Mixed:** $O \subset L$

(i) **Graph:** $P \subset PS \subset L$ (j) **Graph:** $S \subset PS \subset L$ (k) **Graph:** $C \subset O$ (l) **Graph:** $O \subset L$

Figure 11: Comparing update efficiency (in ms) of different propagation chains under different scaling factors

In Figure 11 we illustrate how update times along the different chains are affected by the scaling factors applied to TPC-H, under relational, mixed, and graph semantics. In each scenario, we show the time (in ms) required to perform the operations with respect to the percentage of nodes we update. More specifically, each time reported is averaged over ten runs. For mixed semantics, we use directed edges to ensure referential integrity whenever possible and we only duplicated properties that contribute to foreign keys and are part of the primary key. This ensures the node property key still exists along with the corresponding unique index.

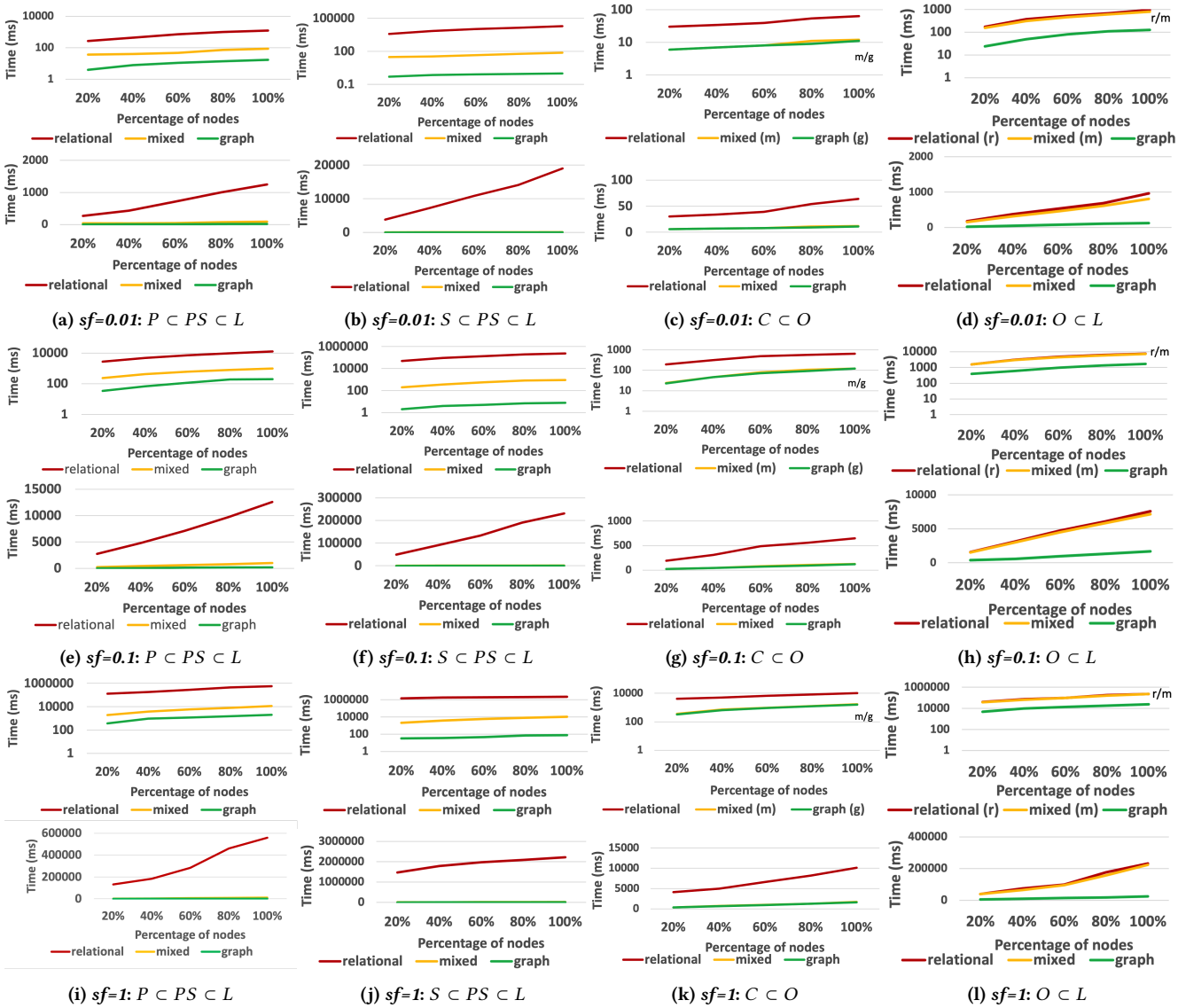
Indeed, mixed semantics represents state-of-the-art support for entity and referential integrity available in graph database systems: we use E/R links whenever possible and rely on duplication of properties otherwise to not lose the ability to enforce the respective node property key. This means, for example, that on nodes labeled Lineitem we still keep the property *orderkey* which is referenced by nodes labeled Orders in an effort to still enforce the key {*orderkey*, *linenumber*} on them. A similar scenario can be observed for PartSupp nodes and the properties *partkey* and *suppkey*.

Under graph semantics, we do not duplicate any properties to enforce referential integrity, but only rely on E/R links. This also means no node property keys nor their unique indices are present. For example, the node key for PartSupp on properties *partkey* and *suppkey* does not exist anymore as these properties now only exist on Part nodes and Supplier nodes, respectively, and they are not duplicated on PartSupp nodes nor on Lineitem nodes. On nodes labeled Part and on nodes labeled Supplier, however, we still enforce the keys for properties *partkey* and *suppkey*, respectively. This means here we can still take advantage of the unique index. Similarly, this holds true for Lineitem nodes where we do not duplicate the property *orderkey* under graph semantics.

It is evident from Figure 11 that update times are directly proportional to the scaling factor, and that for each chain and semantics, there are orders of magnitude performance difference between the scaling factors, consistently for all percentages of nodes considered. In addition, by inspecting one column of Figure 11 at a time, one can see the impact of different semantics on the time as well. While the update performance is as expected for the scaling factors and semantics, the results shown in Figure 11 do quantify these expectations for a diverse range of update maintenance operations.

However, the main questions concerns the impact of our semantics on the update times. For this purpose, Figure 12 is comparing update times between the different semantics directly for each fixed combination of a chain and scaling factor. Due to the big differences in times between the semantics, we show each graph on both a linear and log scale. Noticeably, the performance differs at orders of magnitude, showing the superior handling of referential integrity by our graph semantics over mixed semantics, and that of mixed semantics over relational semantics. The actual differences in performance are direct results of how the data are distributed in the tables and the characteristics of our translations.

For the chain $P_c : P \subset PS \subset L$, performance under mixed semantics is closer to that under graph semantics when compared to the performance difference for the chain $S_c : S \subset PS \subset L$. The simple reason is that there are fewer suppliers than parts but the combinations of suppliers and parts in the PartSupp table are the same. For our chain $C_c : C \subset O$, graph and mixed semantics coincide, so any difference in update performance is negligible. For the chain $O_c : O \subset L$ we duplicate the property *orderkey* on



(a) *sf=0.01*: $P \subset PS \subset L$ (b) *sf=0.01*: $S \subset PS \subset L$ (c) *sf=0.01*: $C \subset O$ (d) *sf=0.01*: $O \subset L$

(e) *sf=0.1*: $P \subset PS \subset L$ (f) *sf=0.1*: $S \subset PS \subset L$ (g) *sf=0.1*: $C \subset O$ (h) *sf=0.1*: $O \subset L$

(i) *sf=1*: $P \subset PS \subset L$ (j) *sf=1*: $S \subset PS \subset L$ (k) *sf=1*: $C \subset O$ (l) *sf=1*: $O \subset L$

Figure 12: Comparing update efficiency of different propagation chains under different semantics

LINEITEM nodes under both relational and mixed semantics. Hence, performance for those semantics are closely aligned.

In Table 6 we have listed the database hits (db hits) and the time (in ms) required for different update chains on the whole TPC-H dataset under different semantics (sem) and for different scaling factors (sf). The database hits already indicate huge performance gains independent of the database size. That is because property redundancy has been reduced in mixed semantics, and eliminated under graph semantics, which the query engine recognizes well. These results emphasize the advantage of referential integrity maintenance that can be harnessed by our proposal to graph modeling using E/R links and E/R keys.

Our experiments quantify the penalty of maintaining referential integrity by the use of property redundancy. Indeed, update

Table 6: Results for update propagation in TPC-H

		sf=0.01		*sf=0.1*		*sf=1*	
chain	sem	db hits	time (ms)	db hits	time (ms)	db hits	time (ms)
P_c	r	1,514,378	1,251	16,315,447	12,579	151,030,378	559,684
	m	42,002	87	420,002	932	4,200,002	11,689
	g	6,001	17	60,001	200	600,001	2,022
S_c	r	29,072,628	18,996	290,157,279	231,131	2,904,674,611	2,216,487
	m	40,102	76	401,002	925	4,010,002	10,584
	g	301	1	3,001	8	30,001	78
C_c	r	106,502	64	1,065,002	648	10,650,002	10,146
	m	4,501	12	45,001	123	450,001	1,707
	g	4,501	11	45,001	122	450,001	1,582
O_c	r	315,877	964	3,152,862	7,606	31,522,479	231,556
	m	315,877	807	3,152,862	7,175	31,522,479	224,179
	g	45,001	125	450,001	1,682	4,500,001	23,516

performance is the worst under relational semantics, which fully utilizes property redundancy. Under mixed semantics, property redundancy is minimized under the constraint to maintain entity integrity by the use of property keys, making it possible to draw on the best support available by current graph database technology. The minimization results in high efficiency gains in many cases. However, the best performance is only possible when property redundancy is eliminated. For this purpose, we need to use E/R keys, which are not currently supported by graph database technology. We iterate that our experiments have been conducted with full use of the unique indices resulting from node property keys. Performance is still magnitudes of order better under graph semantics and without any index support available for E/R keys, highlighting strong prospects for further performance gains once such indices will be developed and implemented.

7 CONCLUSION AND FUTURE WORK

We have proposed E/R graphs as instances of classical E/R diagrams. These constitute a core contribution to conceptual data modeling as E/R graphs are the first graph semantics for the E/R model. Their core advantage is the elimination of attribute/property redundancy through the use of E/R links and E/R keys. Furthermore, we have shown that E/R diagrams are models for hierarchical property graphs, they are property graphs themselves, and their instances, namely E/R graphs, are homomorphisms on their E/R diagram. This constitutes a fundamental contribution to the development of schema languages for property graphs. In fact, E/R diagrams and graphs combine best practice conceptual modeling, logical data modeling, and graph modeling. They constitute the core of PG-schemata that represents well-designed hierarchical database schemata in Inclusion Dependency Normal Form. Our final contribution constitutes fundamental advances in managing entity and referential integrity, revealing a true advantage of graph over other databases. We propose three alternative approaches: graph semantics eliminates property redundancy by using E/R keys, relational semantics fully utilizes property redundancy and only requires property keys, and mixed semantics minimizes property redundancy while still relying on property keys only. Our experiments with the TPC-H dataset firmly quantify how superior graph semantics performs over mixed semantics, and how superior mixed semantics operates over relational semantics. Our results establish E/R keys as an efficiently decidable and core fragment for best data modeling practice among the recently proposed class of PG-keys, which we have shown to be undecidable due to their desired expressiveness. Altogether, the elimination of property redundancy and its resulting benefits appear to be a particular advantage of graph over other types of databases.

Our work motivates the implementation of E/R modeling within graph database systems, including the use of E/R keys and the development of index structures that support their maintenance. Identifying other useful fragments of PG-keys and PG-schemata appears to be another important direction of future research.

REFERENCES

[1] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. 2023. PG-Schema: Schemas for Property Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 198:1–198:25.

[2] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, and Dominik Tomaszuk. 2021. PG-Keys: Keys for Property Graphs. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. 2423–2436.

[3] Renzo Angles and Claudio Gutierrez. 2008. Survey of graph database models. *ACM Comput. Surv.* 40, 1 (2008), 1:1–1:39.

[4] Nimo Beeren and George Fletcher. 2023. A Formal Design Framework for Practical Property Graph Schema Languages. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*. 478–484.

[5] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. 1983. On the Desirability of Acyclic Database Schemes. *J. ACM* 30, 3 (1983), 479–513.

[6] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs.* Morgan & Claypool Publishers.

[7] Peter P. Chen. 1976. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Trans. Database Syst.* 1, 1 (1976), 9–36.

[8] Peter P. Chen. 1997. English, Chinese and ER Diagrams. *Data Knowl. Eng.* 23, 1 (1997), 5–16.

[9] Gwendal Daniel, Gerson Sunyé, and Jordi Cabot. 2016. UMLtoGraphDB: Mapping Conceptual Schemas to Graph Databases. In *Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings*. 430–444.

[10] Victor Martins de Sousa and Luís Mariano del Val Cura. 2018. Logical Design of Graph Databases from an Entity-Relationship Conceptual Model. In *Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2018, Yogyakarta, Indonesia, November 19-21, 2018*. 183–189.

[11] Wenfei Fan and Leonid Libkin. 2002. On XML integrity constraints in the presence of DTDs. *J. ACM* 49, 3 (2002), 368–406.

[12] Wenfei Fan and Ping Lu. 2019. Dependencies for Graphs. *ACM Trans. Database Syst.* 44, 2 (2019), 5:1–5:40.

[13] Ewout Gelling, George Fletcher, and Michael Schmidt. 2023. Bridging graph data models: RDF, RDF-star, and property graphs as directed acyclic graphs. *CoRR* abs/2304.13097 (2023). https://doi.org/10.48550/arXiv.2304.13097

[14] Mark Levene and George Loizou. 2003. Why is the snowflake schema a good data warehouse design? *Inf. Syst.* 28, 3 (2003), 225–240.

[15] Mark Levene and Millist W. Vincent. 2000. Justification for Inclusion Dependency Normal Form. *IEEE Trans. Knowl. Data Eng.* 12, 2 (2000), 281–291.

[16] Heikki Mannila and Kari-Jouko Räihä. 1992. *Design of Relational Databases.* Addison-Wesley.

[17] Jaroslav Pokorný. 2016. Conceptual and Database Modelling of Graph Databases. In *Proceedings of the 20th International Database Engineering & Applications Symposium, IDEAS 2016, Montreal, QC, Canada, July 11-13, 2016*. 370–377.

[18] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71.

[19] Philipp Skavantzos and Sebastian Link. 2023. Normalizing Property Graphs. *Proc. VLDB Endow.* 16, 11 (2023), 3031–3043.

[20] Bernhard Thalheim. 2000. *Entity-relationship modeling - foundations of database technology.* Springer.

[21] Roberto De Virgilio, Antonio Maccioni, and Riccardo Torlone. 2014. Model-Driven Design of Graph Databases. In *Conceptual Modeling - 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*. 172–185.