M. Levene



# The Nested Universal Relation Database Model



Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Author

Mark Levene
Department of Computer Science, University College London
Gower Street, London WC1E 6BT, UK





CR Subject Classification (1991): H.2.1, H.2.3

ISBN 3-540-55493-9 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-55493-9 Springer-Verlag New York Berlin Heidelberg



Typesetting: Camera ready by author/editor
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

# Preface

During the 1980's the *flat relational model* (relational model), which was initiated by Codd in 1970, gained immense popularity and acceptance in the market place. One of the main reasons for this success is that the relational model provides *physical data independence*, i.e. changing the physical organization of the database does not require alteration of the database at the conceptual level. However, the relational model does not provide *logical data independence*, since users must navigate amongst the flat relations in the database when posing queries to the database. Logical data independence would imply that changing the database at the conceptual level does not have an effect on the user's view of the database.

The *universal relation model* (UR model) endeavours to achieve logical data independence in the relational model by allowing the user to view the flat database as if it were composed of a single flat relation. To this end, the user is provided with a UR interface - with all the semantics embedded into the attributes - encapsulating the user's view of the flat database at the external level, on top of the conceptual level. The UR model was firmly established in mainstream relational database theory during the mid 1980's with the introduction of the *weak instance* approach. In the weak instance approach to the UR model, the *representative instance* becomes the underlying data structure of the UR model, which is suitable for storing all the data in the flat database in a single flat relation. Although the application areas of the UR model are slightly restricted by several underlying assumptions it could serve well as the foundation of a natural language interface to a database.

In recent years there has been a growing demand to use databases in non-business applications, such as: office automation, computer aided design (CAD), computer aided software engineering (CASE), image processing, text retrieval, expert systems and geographical and statistical analyses. The flat structure of relations imposed by the first normal form assumption on relational databases poses a severe restriction on the modelling capabilities of the relational model for such non-business applications.

In order to facilitate the modelling of the above non-business applications the *nested relational model* was developed during the 1980's as an extension of the relational model. The nested relational model achieves this wider applicability by allowing hierarchically structured objects, also referred to as *complex objects*, to be modelled directly, whilst maintaining the sound theoretical basis of the relational model.

One of the problems with the nested relational model is that it may prove to be too complex for non-technical users to interact with. This *usability problem* arises because of the fact that queries posed to a nested database involve navigation both amongst and within the structure of nested relations in the nested database. Thus, as in the flat relational model, the nested relational model does not provide logical data independence. Moreover, posing queries to the nested database is much more difficult in the nested

relational model than in the flat relational model due to the hierarchical structure of nested relations.

In this monograph we propose to alleviate the usability problem by providing logical data independence to the nested relational model. To this end we extend the UR model to nested relations by defining a new database model, called the *nested Universal Relation model* (nested UR model). Logical data independence is achieved by allowing users to view the nested database as if it were composed of a single nested relation. Moreover, the nested UR model allows users to interact with the nested database without having to know its structure, which may be complex.

In order to formalize the nested UR model we present a comprehensive formalization of the nested relational model, which incorporates null values into the model. We provide semantics to the nested relational model in terms of *null extended data dependencies*. These dependencies are obtained by extending from flat relations to nested relations both functional data dependencies and the classical notion of lossless decomposition. Furthermore, we define the *extended chase* procedure in order to test the satisfaction of the said null extended data dependencies and to infer more information from a given nested relation. The theory of the nested UR model is established by extending the weak instance approach to the classical UR model to the *nested weak instance* approach to the nested UR model. The nested weak instance approach leads naturally to the definition of the underlying data structure for the nested UR model, namely, the *nested representative instance* (NRI) over the *nested universal relation scheme* (NURS).

A major result of the monograph is that the NRI over the NURS is a suitable model for storing the data in a nested database in a single nested relation. Thus, the classical UR model becomes a special case of the nested UR model. An important implication of this result is that a UR interface can be implemented by using the nested UR model, thus gaining the full advantages of nested relations over flat relations. In particular, redundancy is minimized, query processing becomes more efficient, semantics are often explicitly represented within the nested relations, and we gain more expressive power as both flat and hierarchical data may be presented to the user.

We believe that usability of complex object databases is one of the challenges of the 1990's in the area of database management, as the research into database models for complex objects is gaining maturity. Therefore, both database researchers and practitioners can benefit from the approach of the nested UR model, which is to formally show how one can reduce the complexity of the user interface at the external level of a database in order to gain usability.

This monograph is a slightly revised version of my thesis, which was submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in the University of London in November 1989 and was obtained in June 1990. The thesis was written at the Computer Science Department of Birkbeck College, which is part of the University of London.

I would like to thank my supervisor Professor George Loizou for the many hours he devoted to discussing and carefully reading the thesis. It has been a pleasure for me to work with George who has always had the patience to find and painstakingly correct my mistakes and give me the necessary guidance. I am also grateful to my parents and the rest of the family for giving me the moral and financial support I needed during my studies. In particular, I would like to thank my brother Dan for designing the illustration on the cover of the monograph. I also wish to thank the Wingate Foundation for the financial support that was provided during my last year of study. Finally, I would like to dedicate this monograph to Sara.

London
February 1992

Mark Levene

# Table of Contents

# Chapter 1
# Introduction

This monograph describes a method of data modelling whose basic aim is to make databases easier to use. It presents the *nested Universal Relation model* (nested UR model), which extends the classical UR model to nested relations. In Section 1.1 we give some background material and motivation for defining the nested UR model, and in Section 1.2 we briefly outline the results obtained in the remaining chapters of the monograph.



## 1.1 Background and Motivation of the Monograph

A *database management system* (DBMS) can be viewed via three levels of abstraction: the physical level, the conceptual level (or the logical level) and the external level (or the view level) [ANSI 1975; Ullman 1982a]. The physical level is concerned with the implementation of the database on physical devices. The conceptual level is concerned with the modelling of the database, i.e., how to represent an abstraction of the real world. Finally, the external level is concerned with the user's view of the database, i.e., it is an abstraction of the conceptual database.

The three DBMS levels can provide two levels of *data independence* via a database model. A database model provides *physical data independence* if changing the physical organization of the database does not require alteration of the database at the conceptual level. On the other hand, a database model provides *logical data independence* if changing the database at the conceptual level does not have an effect on the user's view of the database.

One of the main reasons the *flat relational model* (relational model) [Codd 1970, 1979] has recently gained immense popularity and acceptance in the market place is that it provides physical data independence. However, the relational model does not provide logical data independence, since users must navigate amongst the relations in the database when posing queries to the database. It, therefore, follows that changes at the conceptual level of the database will necessitate changes to the queries posed to the database, and thus application programs may be impaired.

The *universal relation model* (UR model) [Ullman 1982b, 1983a; Maier & Ullman 1983; Sagiv 1983; Maier et al. 1984, 1986; Mendelzon 1984; Brosda & Vossen 1988] endeavours to achieve logical data independence in the relational model by allowing the user to view the database as if it were composed of a single relation. To this end, the user is provided with a UR interface [Ullman 1983a] - with all the semantics embedded into the attributes - encapsulating the user's view of the database at the external level, on top of the conceptual level. It is important to point out that a database application using the

UR model must satisfy several assumptions, hereafter called the *UR assumptions* [Maier et al. 1984, 1986]. The most fundamental UR assumption is the *universal relation scheme assumption*, which states that there is a universal set of attributes, U, for the application being modelled, and each attribute in this universal set, U, has a unique meaning. It is also assumed that every set of attributes, $X \subseteq U$, has a *basic relationship* which results in a flat relation over X, denoted as [X], and called the *window* for X [Maier et al. 1986].

The theory of the UR model was firmly established in the mid 1980's with the introduction of the *weak instance* approach [Honeyman 1982; Maier et al. 1984; Mendelzon 1984; Graham et al. 1986; Atzeni & Bernardis 1987]. In the weak instance approach to the UR model, the *representative instance* (RI) [Sagiv 1981, 1983; Maier et al. 1984; Mendelzon 1984] becomes the underlying data structure of the UR model, which is suitable for storing all the data in the database in a single relation.

A database model consists of three main components, which we now enumerate:

(1) The data structures of the model.

(2) The query language of the model.

(3) The integrity constraints of the model.

In the relational model the data structures are relations (also referred to as *flat relations*), the query language is the relational algebra, and the integrity constraints are data dependencies [Codd 1979; Ullman 1982a; Maier 1983].

In recent years there has been a growing demand to use databases in non-business applications, such as: office automation, computer aided design (CAD), image processing, text retrieval, expert systems and geographical and statistical analyses [Scholl & Schek 1987; Abiteboul et al. 1989b]. The *nested relational model* [Makinouchi 1977; Roth et al. 1985, 1988; Abiteboul & Bidoit 1986; Schek & Scholl 1986; Thomas & Fischer 1986; Ozsoyoglu & Yuan 1987a; Van Gucht & Fischer 1988; Levene & Loizou 1989a] was developed as an extension of the relational model in order to facilitate the modelling of the above non-business applications. The nested relational model achieves this wider applicability by allowing hierarchically structured objects, also referred to as *complex objects*, to be modelled directly, whilst maintaining the sound theoretical basis of the relational model.

Some of the advantages of nested relations in comparison to flat relations are now mentioned. Nested relations minimize redundancy of data, and allow efficient query processing since some of the joins are realized within the nested relations themselves. In addition, nested relations allow explicit representation of the semantics of the application within their structures, and provide a more flexible user interface, which allows both flat and hierarchical data to be presented to the user.

One of the problems with the nested relational model is that it may prove too complex for non-technical users to interact with. This *usability problem* arises because of the fact that queries posed to a nested database involve navigation both amongst and within the structure of nested relations in the nested database. Thus, as in the relational model, the nested relational model does not provide logical data independence. Moreover, posing queries to the nested database is much more difficult in the nested relational model than

in the flat relational model due to the hierarchical structure of nested relations. The usability problem escalates even further when we take into account the application programs which may be impaired because of changes to the nested database at the conceptual level.

In this monograph we propose to alleviate the usability problem by providing logical data independence to the nested relational model. To this end we extend the UR model to nested relations by defining a new database model, called the *nested Universal Relation model* (nested UR model).

## 1.2 Outline of the Monograph

We now outline our presentation of the nested UR model as given in the monograph. The preliminary material needed to develop the nested UR model is given in Chapter 2, wherein we present the underlying database models. In Section 2.1 we describe the flat relational model, in Section 2.2 we present the nested relational model, and finally in Section 2.3 we describe the classical UR model.

In order to formalize the nested UR model, we first define the *null extended nested relational model* in Chapters 3 and 4. Nulls [Biskup 1981; Zaniolo 1984; Roth et al. 1985; Codd 1986, 1987; Levene & Loizou 1989a, 1989c] play an important role in the nested UR model, and thus the null extended nested relational model is essentially a comprehensive extension of the nested relational model so as to include nulls.

In Section 3.1 we define null extended nested relations (from now on referred to simply as nested relations) over null extended domains, which are the domains of nested relations that may include null values. We then define the *null extended algebra* in Section 3.2, which is a *complete* extended algebra for manipulating nested relations (cf. [Abiteboul et al. 1989a]). The motivation for defining the null extended algebra is that previously defined extended algebras for nested relations [Jaeschke 1985a, 1985b; Roth et al. 1985, 1988; Abiteboul & Bidoit 1986; Schek & Scholl 1986; Thomas & Fischer 1986; Deshpande & Larson 1987; Gyssens 1987; Houben & Paredaens 1987; Colby 1989; Levene & Loizou 1989a] are not suitable for the nested UR model. This is mainly due to the fact that in order to formulate queries with the aforementioned extended algebras the structure of the nested relations in the nested database needs to be known, whilst the null extended algebra, presented in this monograph, frees the user from navigation within the individual nested relations in the nested database. In addition, the extended (natural) join operators defined in the aforementioned extended algebras have been very limited, which is a major drawback in a UR environment. Finally, the outer join operator [Codd 1979; Date 1987b] and the total projection operator [Maier et al. 1984], which are essential to the UR model, have not so far been extended to nested relations. Thus in order to solve these problems, the null extended algebra provides us with the general *null extended join* operator, the *null extended outer join* operator, and the *null extended total projection* operator.

In Sections 4.1 to 4.4 we define semantics for the null extended nested relational model in terms of *null extended data dependencies*, which are integrity constraints over

nested relations.

Although there is a growing body of work on the nested relational model, data dependency theory for nested relations has mainly concentrated on extending the functional dependency (FD) to nested relations [Makinouchi 1977; Jaeschke & Schek 1982; Arisawa et al. 1983; Fischer et al. 1985; Miura et al. 1986, 1987; Van Gucht & Fischer 1986, 1988]. The class of *null extended data dependencies* which we define for the null extended nested relational model includes: *null functional dependencies* (NFDs), *null extended functional dependencies* (NEFDs), and *null extended join dependencies* (NEJDs). NFDs provide a redefinition of FDs so as to include null values in the context of nested relations, NEFDs provide an extension of NFDs to nested relations, thus allowing relation-valued attributes to appear in NFDs, and finally NEJDs provide an extension of *join dependencies* [Beeri & Vardi 1981; Sciore 1982] so as to include null values in the context of nested relations.

The important notion of a *lossless decomposition* [Ullman 1982a; Maier 1983] from relational database theory has, so far, not been extended to nested relations; it has rather been used in nested relational theory whenever nested relations support a flat relational interface [Kambayashi et al. 1983; Ozsoyoglu & Yuan 1987a, 1987b; Kambayashi & Yamamoto 1987; Scholl et al. 1987]. We fill in this gap, via NEJDs, by defining the novel notion of a *null extended lossless decomposition*, thus extending the classical notion of a lossless decomposition to nested relations. Furthermore, we conjecture that null extended data dependencies are sufficient to model most real-world applications (cf. [Fagin 1982; Beeri & Kifer 1986]).

In Section 4.5 we define the *extended chase* procedure (also referred to simply as the extended chase), which extends the classical *chase* procedure [Maier et al. 1979; Beeri & Vardi 1984; Graham et al. 1986] to nested relations. The extended chase allows us to test the satisfaction of a set of null extended data dependencies and to infer more information from a given nested relation. Thus, the extended chase provides both a theorem prover and an inference engine for the null extended nested relational model.

**Example 1.1.** Schemas of nested relations are represented by *scheme trees* [Ozsoyoglu & Yuan 1987a], as shown in Figure 1.1. The *nested relation scheme* (NRS), for the scheme tree, T, denoted by R(T), is: TUTOR SALARY (CHILD)* (DAY)*, where attributes with relation-valued domains are marked by *, in order to distinguish them from attributes defined over simple domains [Abiteboul & Bidoit 1986; Ozsoyoglu & Yuan 1987a]. A nested relation, r*, over the NRS, R(T) = TUTOR SALARY (CHILD)* (DAY)*, for the scheme tree, T, of Figure 1.1, is shown in Figure 1.2. We note that *null* in r* denotes a null value in the nested relation. The semantics of null values are such that *null* stands for either: an unknown value, a non-existent value, or a no-information value which may be unknown or non-existent.

The semantics of the nested relation, r*, over the NRS, R(T), are captured by the following set of null extended data dependencies, namely, the NFD: TUTOR $\rightarrow$ SALARY, and the NEFDs: TUTOR,SALARY $\rightarrow$ (CHILD)* and TUTOR,SALARY $\rightarrow$ (DAY)*. In other words, a TUTOR, who has a unique SALARY, has a unique set of

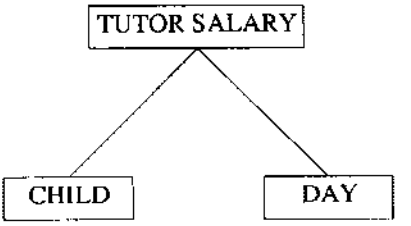


Fig. 1.1. The scheme tree T.

CHILDren and gives tutorials on a unique set of DAYs.

| TUTOR | SALARY | (CHILD)* | (DAY)* |
|---|---|---|---|
| | | CHILD | DAY |
| Robert | 12000 | Hanna<br>Brian | Monday<br>Thursday |
| Hanna | 14000 | Annette<br>Ada | *null* |
| Martine | *null* | *null* | *null* |
| *null* | 15000 | *null* | Wednesday |
| *null* | *null* | Ruth | Tuesday<br>Friday |

Fig. 1.2. The nested relation, r*, over the nested relation scheme, R(T).

After having defined the null extended nested relational model, we have at our disposal the tools needed to formalize the nested UR model. This formalization is presented in Chapter 5 wherein we define the general case of the nested UR model for a nested database.

The theory of the nested UR model is vigorously established in Sections 5.1 and 5.2 by extending the weak instance approach to the UR model to the *nested weak instance* approach to the nested UR model, under a set of null extended data dependencies for a nested database. This leads us to define the underlying data structure of the nested UR model, namely, the *nested representative instance* (NRI), over the *nested universal relation scheme* (NURS), which allows us to model the semantics of the nested database within a single nested relation. The NRI extends the RI to nested relations, while the NURS provides the necessary NRS over which nested relations in the nested database can be joined automatically by using the null extended join operator, in order to provide the desired logical data independence referred to earlier.

In Section 5.1 we discuss in detail how to obtain the NURS for a given nested database. To this end we provide two restructuring operators on scheme trees and one restructuring operator on the corresponding nested relations in the nested database. We then give

an algorithm for obtaining the NURS and we prove its correctness.

In Section 5.2 we define the nested weak instance approach to the nested UR model with its associated NRI under a set of null extended data dependencies for a nested database. Under the nested weak instance approach to the nested UR model, the window, [X], for any $X \subseteq U$, is taken to be the unnesting of the null extended total projection of the NRI onto X. We then proceed to show that the NRI can always be constructed by invoking the extended chase given in Chapter 4.

In Section 5.3 we present one of the major results of the monograph, i.e., showing that the NRI, over the NURS, is a suitable model for the data to be stored in a single nested relation, given a nested database and a set of null extended data dependencies, for any application satisfying the UR assumptions. In other words, the NRI encapsulates all the information in the nested database within a single nested relation satisfying the set of null extended data dependencies. An important implication of this result is that a UR interface providing both flat and hierarchical outputs can be implemented under the nested UR model. Thus, we can gain the advantages of nested relations compared to flat relations via the implementation of a UR interface; therefore, the usability problem can be successfully solved. Furthermore, the classical UR model under the weak instance approach is shown to be a special case of the nested UR model under the nested weak instance approach. We also show that the nested UR model is strictly more expressive than the UR model, since nested relations are strictly more expressive than their flat counterparts [Miura et al. 1986]. This fact implies that the range of applications that can be modelled under the nested UR model is much larger than the corresponding range of applications that can be modelled via the UR model.

The NRI can always be constructed via the extended chase. The extended chase is a very useful theoretical tool; however, constructing the NRI via the extended chase is, in general, computationally inefficient. In addition, the extended chase is unlikely, in the near future, to be supported by a DBMS supporting the null extended nested relational model. Thus, in Section 5.4, we investigate a computational approach to the nested UR model as a special case of the nested weak instance approach to the nested UR model (cf. [Maier et al. 1984]). Our computational approach to the nested UR model is based on the *UMC property*, which we now very briefly discuss.

Results from UR theory have shown a strong connection between $\gamma$-acyclic databases and the UR model [Yannakakis 1981; Fagin 1983; Chan & Atzeni 1985; Biskup et al. 1986; Jajodia & Springsteel 1987]. In particular, if a database is $\gamma$-acyclic then there is a unique join sequence for computing queries over any subset of the universal set of attributes. In addition, we have in $\gamma$-acyclic databases a *unique minimal connection* (UMC) amongst any subset of the universal set of attributes. This property, called the *UMC property*, is defined and extended to nested databases in Chapter 2.

In Section 5.4 we utilize the UMC property in the definition of an algebraic construction of the NRI via the null extended outer join operator of the null extended algebra. In addition, we show that, given the UMC property, query processing over any subset of the universal set of attributes can be done algebraically, by defining a *window function* [Maier et al. 1986] to compute [X] for any $X \subseteq U$.

In order to define the DBMS levels of the nested UR model, we depart from the traditional three-level architecture of a DBMS by adding a fourth level, called the *internal level*, between the physical level and the conceptual level. At the physical level of the nested UR model we have the physical database, which we do not discuss any further in this monograph. At the internal level we have the null extended nested database (which we have simply referred to as the nested database), while at the conceptual level we have the NRI over the NURS. Finally, the nested UR model supports a UR interface at the external level, within which the user may view the data in either a flat or a hierarchical fashion.

In order to summarize the DBMS levels of the nested UR model we show in the diagram of Figure 1.3 the differences between the DBMS levels of the classical UR model and those of the nested UR model.

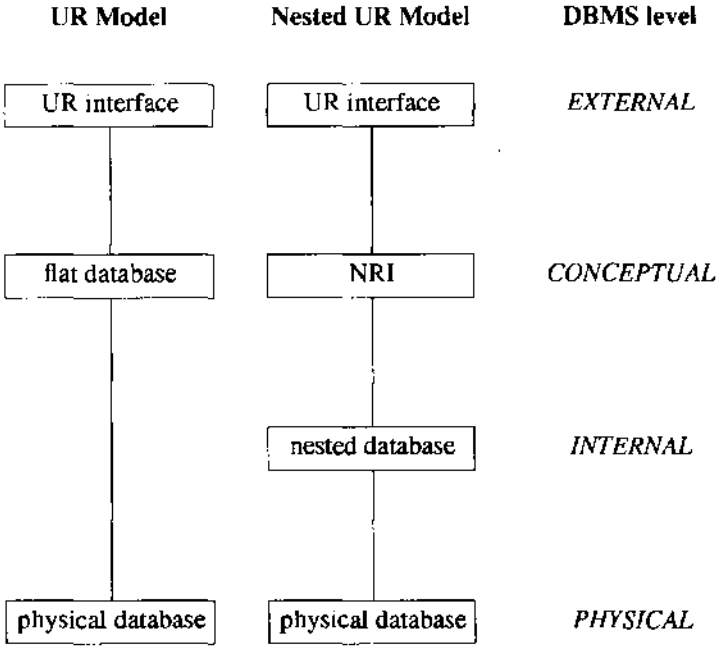


Fig. 1.3. Comparison between the DBMS levels in the classical and the nested UR models.

An important special case of the nested UR model is when we only have a single nested relation in the nested database at the internal level. In this case the internal and conceptual levels of the nested UR model coincide, and we thus obtain the traditional three-level DBMS architecture. The situation of having a nested UR model with a single nested relation is ideal for query processing, since all the null extended joins are realized within this nested relation.

In Chapter 6 we investigate several ways of viewing the nested UR model comprising a single nested relation and show that these different approaches coincide, when we view the single nested relation as the NRI under a set of null extended data dependencies.

Firstly, we show, in Section 6.1, that a nested relation can be viewed in terms of the *association-object database model* [Maier & Warren 1982; Maier et al. 1986, 1987]. Thus, for the scheme tree, T, shown in Figure 1.1, we have the associations: {{TUTOR, SALARY}, {TUTOR, SALARY, CHILD}, {TUTOR, SALARY, DAY}}, and the object: {TUTOR, SALARY, CHILD, DAY}.

Another way of viewing a nested relation is in terms of γ-acyclic databases. Thus, in Section 6.2, we show that nested relations correspond to a subclass of γ-acyclic databases [Levene & Loizou 1989d]; this implies a strong connection between nested relations and the UR model [Biskup et al. 1986]. In fact, for the scheme tree, T, shown in Figure 1.1, we have the induced γ-acyclic database scheme: {{TUTOR, SALARY, CHILD}, {TUTOR, SALARY, DAY}}.

In Section 6.3 we discuss the effects on the NRI, when we assume a nested database composed of a single nested relation satisfying a set of null extended data dependencies. When viewing the single nested relation as the NRI, over the NURS, we obtain all the desirable properties discussed in Chapter 5, and consequently the NRI is fully optimized. In particular, query processing can be effected algebraically, since the window, [X], for any $X \subseteq U$, is now simply the unnesting of the null extended total projection of the single nested relation onto X. Finally, in this special case all of the UR assumptions are automatically satisfied in the nested relation. In order to bring out the advantages of the nested UR model, in this special case of a single nested relation, we give some application examples in Section 6.4.

**Example 1.2.** Let d* = {r*} be a nested database for the single nested relation, r*, over the NRS, R(T), of Example 1.1, and let D* = {TUTOR → SALARY, TUTOR,SALARY → (CHILD)*, TUTOR,SALARY → (DAY)*} be the set of null extended data dependencies that hold in r*. Then r*, over the NRS, R(T), is the NRI under the set of null extended data dependencies, D*, for the nested database d*.

Finally, in Chapter 7 we conclude the monograph with some final remarks and discuss ongoing research resulting from the formalization of the nested UR model.