ON CONSISTENT EXTENSIONS TO THE RELATIONAL DATABASE MODEL

James Clifford and Albert Croker

December 1986

Center for Research on Information Systems
Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #135

GBA #86-100

Table of Contents

List of Figures

1. Introduction

In this paper we formally define the concept of a *consistent extension* to the relational data model. We use this concept to define the properties of the relational data model that should be retained in any extension to that model.

Since its introduction the *relational data model* [Codd 70] has gained wide acceptance among users of database management systems (DBMS). In part, this acceptance is due to the simplicity of its data structure, the relation, and the formalism used to define this structure and the operations used to manipulate it, the relational algebra (or calculus).

In spite of, or more likely, because of its acceptance certain limitations of the relational data model have become apparent. A database can be viewed as a model of some aspects of the world. Although adequate as a modelling tool, the relational data model sometimes forces an "unnatural" representation of certain of these real-world aspects. That is, use of the relational model does not always permit the representation of what in the real world is perceived as a single object (entity or relationship in the *entity-relationship data model* [Chen 76]) as a single tuple in one relation. We feel that such a representation would be more natural in that it more closely coincides with our conceptual view of an object as being a single collection of the properties which serve to define it, and improve the conceptual modeling capabilities of the relational data model.

Various proposals have been made to extend the relational model to remedy certain of these limitations (for example, [CliffordCroker 87], [JaeschkeSchek 82], [Katz 86], [OsbornHeaven 86], and [StonebrakerRowe 86]). As extensions to the relational data model each of these proposals retains some of the properties associated with the relational data model, while modifying and/or adding additional ones. Among the properties associated with the relational data model are: the structure of a relation; the set of (algebraic) operations (union, intersection, complementation, project, join and select); and the algebraic properties (commutativity, associativity, distributability, etc.) of these operations; and the class of (expressable and enforceable) constraints (typically Functional Dependencies (FDs) and Multi-Valued Dependencies (MVDs).

Usually proposals to extend the relational data model have required that the structure of relations be modified into what is generally called non-first normal form relations. When modifying the structure of a relation, it is often necessary to redefine the operations that operate on these structures. In turn, these new definitions can change the properties that were associated with these operations. The final result may be an extension that is inconsistent and differs in unexpected ways from the relational model.

Since much of the power of the relational model comes from its being **well-defined** in the mathematical sense, it is somewhat alarming to contemplate proliferating proposals to "extend" it that in fact are proposals for a **different** model altogether. In the spirit of providing a "yardstick" to determine whether a proposed "extension" is in fact consistent with the relational model and a true extension, we propose a definition of a **consistent extension to the relational data model**. This concept that we define can be viewed as a "general" relational data model template that can be instantiated in different ways to achieve relational-like data models with differing modelling capabilities. Similar in spirit to the classification of **implementations** of the relational model as "tabular," "minimally relational," "relationally complete," and "fully relational," ([Codd 82], [Date 86]) the concept of a consistent extension addresses itself to the **model** itself, and what are its essential components.

The paper is organized as follows. We begin in Section with a formal description of the relational model, and other related definitions which we will make use of in succeeding chapters. In Section we formally define the notion of a consistent extension to the relational data model, defining consistent extensions for relations, the relational algebra, and discussing the impacts on some of the constraints associated with the relational data model. We conclude in Section summarizing what we have done, and describing future directions of this research.

2. Definition/Notation

In this section we specify the relational data model in terms of the data structures used for organizing data, relations, and the operations for manipulating these data structures. Traditionally the set of operations for manipulating relations has been specified using one of two equivalent formulations, the relational algebra or the relational calculus [Maier 83]. In this paper we will focus on the relational algebra.

The Relation:

Let the set U be a universe of attributes, where an attribute is just the name of some property of interest (e.g. **NAME, ID#, PRICE**, etc.).

Associated with each attribute $A_i \in U$ is a set of atomic (non-decomposable) values D_i called the domain of attribute A_i and denoted $dom(A_i)$

Let $\mathbf{A} = \{A_1, A_2, ..., A_n\}$ be a set of attributes and $\mathbf{D} = dom(A_1) \cup dom(A_2) \cup ... \cup dom(A_n)$ be the union of the corresponding domains.

A *relation* r over the set of attributes **A** is a finite set of mappings (*tuples*) $r(\mathbf{A})$ where

$$r(\mathbf{A}) \subseteq \{t \mid (t: \mathbf{A} \text{->} \mathbf{D}) \;\&\; (t(A_i) \in dom(A_i))\}$$

We use the notation $t(A_i)$ to denote both the value derived by applying t to A_i, and the function derived by restricting t to A_i. Thus for a set of attributes **X** of size n, $t(\mathbf{X})$ is either an n-tuple of values, or the function resulting from restricting t to **X**.

Relations may be conveniently represented as a table. For example, a relation r on the set of attributes $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ can be represented as in Figure 2-1, where each row represents a tuple, and each value in the row results from applying the tuple to the attribute that heads the column containing that value. For example, if t_3 is the tuple represented by the third row, then $t_3(\mathbf{A}) = a_1$, $t_3(\mathbf{B}) = b_3$, and $t_3(\mathbf{C}) = c_2$.

A1	A2	A3
a1	b1	c1
a2	b2	c1
a1	b3	c2
a3	b2	c3

Figure 2-1: Relation as Table

A *relation scheme* $\mathbf{R} = <\mathbf{A}, \mathbf{K}>$ is a two-tuple where

- $\mathbf{A} = \{A_1, A_2, ..., A_n\} \subseteq \mathbf{U}$ is a set of attributes,

- $\mathbf{K} \subseteq \mathbf{A}$ is the *designated key* of the relation scheme, and

A relation r on relation scheme $\mathbf{R} = <\mathbf{A}, \mathbf{K}>$ is a finite set of tuples $\{t_1, \ldots, t_n\}$ such that

- $\mathbf{r}$ is defined over the set of attributes $\mathbf{A}$, and
- $\forall t_i \in \mathbf{r}, \forall t_j \in \mathbf{r} \; [t_i(\mathbf{K}) \neq t_j(\mathbf{K})]$.

2.1. The Relational Algebra

Relations are sets and as such they can be manipulated by the standard set operations of *union*, *intersection*, and *complementation*. However, since all of the tuples of a relation must be defined over the same set of attributes, the operand and result relations to set operations must all be pairwise *union-compatible*. Two relations are *union-compatible* if their tuples are defined over the same sets of attributes. The complement of a relation $\mathbf{r}$ is defined relative to the relation containing all possible tuples defined over the same set of attributes as are the tuples in $\mathbf{r}$.

The remaining relational algebra operations that we will discuss are defined in terms of attribute values (*selection* and *join*), or can be used to construct a relation that has a set of tuples defined over a different set of attributes that its operands (*project* and *join*.)

Let r be a relation over the set of attributes $\mathbf{A} = \{A_1, A_2, \ldots, A_m\}$ and s be a relation over the set of attributes $\mathbf{B} = \{B_1, B_2, \ldots, B_n\}$ where $\mathbf{A} \cap \mathbf{B} = \phi$.

Project

The *projection* of the relation r on to the set of attributes $\mathbf{X}$ (denoted $\Pi_{\mathbf{X}}(r)$) where $\mathbf{X} \subseteq \mathbf{A}$, is the relation over the set of attributes $\mathbf{X}$, defined as

$$\Pi_{\mathbf{X}}(r) = \{t(\mathbf{X}) \mid t \in r\}$$

Select

The selection from r of those tuples satisfying $t(A_i) \; \theta \; a$ (denoted $\sigma_{A_i \, \theta \, a}(r)$) where θ is a comparator, is the relation over $\mathbf{A}$ defined as

$$\sigma_{A_i \, \theta \, a}(r) = \{t \mid t \in r \wedge t(A_i) \; \theta \; a\}$$

Join

The join of two relations r and s (denoted $r \underset{A_i \theta B_j}{|X|} s$) is the relation over the set of attributes $\mathbf{A} \cup \mathbf{B}$ formed by combining a tuple from r with tuple from s whenever the A_i value of the tuple from r stands in a θ relationship with the B_j value of the tuple from s.

$$r \underset{A_i \theta B_j}{|X|} s = \{t \mid \exists\, t_r \in r, \exists\, t_s \in s \wedge t(\mathbf{A}) = t_r \wedge t(\mathbf{B}) = t_s\}$$

3. Definition of a Consistent Extension

A data model can successfully be regarded as a three-tuple <S,O,C> consisting of its structures, its operations, and its allowable constraints [Codd 81]. These three components of the relational model will be discussed in turn before we define the notion of a consistent extension to the relational data model.

The following definition makes precise exactly what we mean by **the relational model**:

Definition: By *the relational data model* or ***RDM*** is meant that structure <***1NF***, ***RA***, ***C***> consisting of:

1. ***1NF***, the class of first normal form relations,
2. ***RA***, the relational algebra ($\cup$, $\cap$, -, σ, Π, $|X|$)
3. ***C***, the class of FDs and MVDs

3.1. The Structures: Relations

In the definition of the relational data model the domain associated with relation attributes are restricted to simple (atomic) values. We refer to these as *base domains*. That is, for any attribute $\mathbf{A}$ in a relation scheme $\mathbf{R}$, and t, a tuple in a relation r defined over $\mathbf{R}$, $t(\mathbf{A})$ is treated as a non-divisible value. All such relations are said to be in in ***first normal form*** (***1NF***).

Although this restriction, which we will call the *base domain constraint*, was originally intended to ease the task of implementing relational database management systems [Codd 70], it has had the added affect of increasing the level of detail needed to model many of the types of objects and events commonly occurring in applications for which database management systems are used. For example, in a database containing data on employees it might be necessary to maintain a list of dependents for each employee. Since an employee can have an arbitrary number of dependents, it is generally not feasible to represent all of the employee data in a single relation where there is a one-to-one correspondence between employees and tuples.

Figure 3-1 shows one approach that would likely be taken to represent employee data under the base domain constraint. In this approach two relations schemes are used. One, scheme **EMP**, is used to group those attributes for which each employee is assumed to have only one (possibly null) value. The second scheme, **EMP-DEP** is used to relate an employee with each of his or her dependents. A relation defined over this scheme will have one tuple for each dependent of each employee.

EMP ($\underline{\text{emp\#}}$, emp-name, dept#, ...)

EMP-DEP ($\underline{\text{emp\#, dep-name}}$, ...)

Figure 3-1: Example Relation Schemes

A second example arises for those instances when it is desirable to maintain *historical* data in a database so that **previous**, as well as **current** values of designated attributes are retrievable. For example, it might be necessary to retain data about what department an employee was in at some point in the past, or what his or her salary was at that time.

In this situation it is likely that some variation of the scheme shown in Figure 3-2 would be used. With this approach an additional attribute, **time**, (with an appropriate domain) is somehow added to the relation scheme. For each employee the relation contains one tuple for each time period in which a value is to be maintained. The attribute **time** is used to indicate the period for which the values represented by the tuple are relevant.

EMP ($\underline{\text{emp\#}}$, $\underline{\text{time}}$, emp-name, dept, ...)

Figure 3-2: Example Historical Relation Scheme

The *relation-like* data models that have been proposed, have usually attempted to increase the

expressiveness of the relational model while retaining many of the properties of this model. As such the proposed data models are best viewed as extensions of the relational model. In almost all cases, the proposed data models differ from the relational data model in their relaxation of the restriction that domain values be atomic. This difference is the source of their increased expressiveness.

By relaxing the base domain constraint it is possible to define a greater variety of relations than would otherwise be possible. Relaxing this constraint implies the introduction of some kind of *structured domain* into the model; since this can be done in different ways it is possible to define different extensions to ***1NF*** relations. Each set of *non-first normal form* (***N1NF***) relations that is defined by each of these extensions subsumes the set of ***1NF*** relations.

In order to view the set of ***1NF*** relations and a specific extension to them in a consistent fashion it will often be necessary to express the domains of ***1NF*** relations in an "extension-dependent" way. For example, if a given extension defines domains as consisting of values that are sets, then the domain values of ***1NF*** relations can be expressed as being restricted to single-valued sets as shown in Figure 3-3(a). Similarly for an extension that defines domains as consisting of values that are functions from one underlying primitive domain to another, the domain values of ***1NF*** relations can be expressed in terms of constant functions, as in Figure 3-3(b). (Function-valued domains can be used to represent the *3-dimensional* relations that have been proposed to represent historical relational data models.)

We formalize this notion of structural extensibility in the following definition.

Definition: Let ***1NF*** be the set of all ***1NF*** relations and let ***N1NF*** be the set of all relations definable under a specific relaxation of the base domain constraint. The set of relations ***N1NF*** is a *consistent structural extension* of ***1NF*** if there exists a one-to-one function ψ_S:***1NF*** --> ***N1NF***. (This relationship is shown in Figure 3-3.)

With this definition, a consistent structural extension of ***1NF*** has a representation capability that is at least as great as that of standard relations. That is, if we equate tuples to real world objects, a consistent extension of ***1NF*** relations can represent all of the sets of objects (relations) representable by ***1NF***, and

	A	B	C			A	B	C
r	a1	b1	c1	----->	r'	{a1}	{b1}	{c1}
	a2	b2	c2			{a2}	{b2}	{c2}
	a3	b3	c3			{a3}	{b3}	{c3}

a

	A	B	C	----->		A	B	C
r	a1	b1	c1		r''	f_{a1}	f_{b1}	f_{c1}
	a2	b2	c2			f_{a2}	f_{b2}	f_{c2}
	a3	b3	c3			f_{a3}	f_{b3}	f_{c3}

where $\forall\, \phi, \forall\, x\; [f_{\phi}(x) = \phi]$

b

Figure 3-3: Two Mappings From *1NF* to Set-Valued Relations

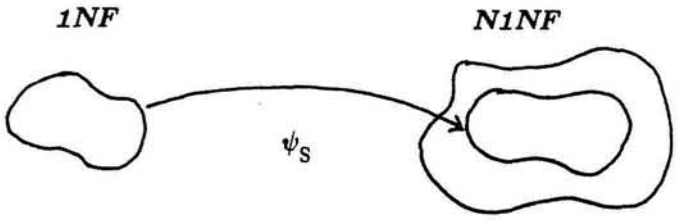

Figure 3-4: Consistent Structural Extensions

possibly sets of more "complex" objects.

3.2. The Operations: Relational Algebra

Relations represent only one component of the relational data model. The second component of the model is the set of operations used to manipulate relations. If a data model is to be considered an extension of the relational data model, then it is reasonable to expect that it contains some counterpart to each of the relational operators. Since a consistent extension to ***1NF*** relations need not contain 1NF relations it is often necessary or desirable to extend the set of relational operators so that they are better matched to the extended set of relations. Extending the set of relational operators may include the redefining of existing operators and/or the defining of additional operators.

For example, let ***FRDM*** be an extended relational data model in which attributes are defined over function-valued domains. We can redefine the selection relational operator so that it better accommodates this extended relation. One possible definition is the following:

$$\sigma_{\mathbf{A}=\mathbf{a},x}(\mathbf{r}) = \{\ t \mid t \in \mathbf{r} \wedge t(\mathbf{A})(x) = \mathbf{a}\}$$

(The resulting relation is defined over the same set of attributes as relation **r**.) This modified operator selects from an extended relation **r** those tuples for which the function represented by their **A**-attribute yields a value of **a** when applied to the value x.

It is also possible to define operators for this model for which, seemingly, there are no standard relational counterparts. As an example, let **r** be an extended relation in ***FRDM*** defined over the set of attributes **R**, and consider the operator ρ:

$$\rho_x(\mathbf{r}) = \{\ t \mid \exists\ t_1 \in \mathbf{r}, \forall\ \mathbf{A} \in \mathbf{R}\ [t(\mathbf{A})(x) = t_1(\mathbf{A})(x) \wedge \forall\ y \neq x\ [t(\mathbf{A})(y)\ \textit{is undefined}]]\}$$

This operator restricts each tuple t of the operand relation **r** so that the function value of each attribute is defined over a single value x. (The ***timeslice*** operator in the Historical Relational Data Model [CliffordCroker 87] is one expression of this operator.)

Using the concept of a consistent structural extension of a set relations, we next define a similar concept for relational operators. Here our goal is to retain the basic mathematical properties of the relational data model while increasing the functionality of the extended model.

A consistent extension to the set of ***1NF*** relations may contain some relations for which there are no ***1NF*** counterparts. However, by definition there must exist a one-to-one function ψ_S that maps each 1NF relation r into a unique relation $\psi_S(r)$ in the consistent extension. We say that any two such corresponding relations are *equivalent under* ψ_S denoted $r \equiv_{\psi_S} \psi_S(r)$.

If a given extension to the relational data model defines a set of relations that are a consistent structural extension to the set of ***1NF*** relations, then it should also extend the definition of the relational operators in a consistent way. That is, these extended operators should behave on the extended structures essentially as the original operators behave in the original model.

More formally, let ψ_S be a relation extension operator that maps ***1NF*** relations into some consistent extension set of relations ***N1NF***. Let θ be an n-ary relational operator where

θ: $1NF_1 \times 1NF_2 \times \ldots \times 1NF_n \text{ --> } 1NF$

(For the standard set of operators discussed in Section n is 1 or 2.) Let θ_E be an m-ary operator where $m \geq n$ and

θ_E: $N1NF_1 \times N1NF_2 \times \ldots \times N1NF_n \times \alpha \text{ --> } N1NF$

That is, θ_E is a mapping from n N1NF relations, and $m - n$ (possible zero) other operators (indicated by α).

Definition: The operator θ_E is a *consistent extension under* ψ_S of the relational operator θ if for each n-tuple $<r_1, r_2, ..., r_n>$ of ***1NF*** relations, $\psi_S(\theta(r_1, r_2, ..., r_n)) = \theta_E(\psi_S(r_1), \psi_S(r_2), ..., \psi_S(r_n), \alpha)$.

This is illustrated in Figure , which illustrates the homomorphism between ***RDM*** and the consistent structural extension under ψ_S. We note further the isomorphism between ***RDM*** and the *image* of ***RDM*** under ψ_S.

Since an extended relational operator can have a greater number of arguments than its relational counterpart, the above definition allows a set of consistent extensions to each relational operator.

Definition: A set of operators θ is a *consistent operator extension* of the relational algebra **RA** if θ includes a consistent extension of each of the relational operators.

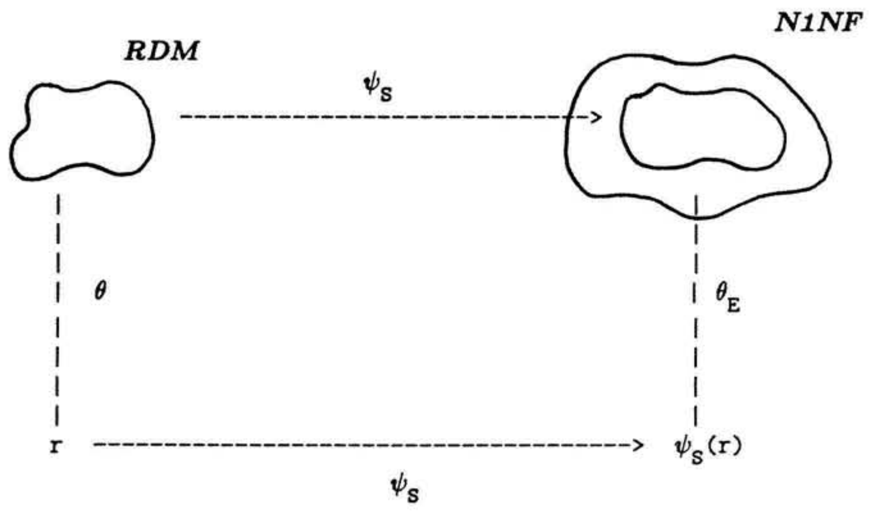

Figure 3-5: ψ_S: A Homomorphism Between ***RDM*** and ***N1NF***

3.3. The Constraints: FDs and MVDs

A database constraint *c* is a statement in a suitable language which restricts the allowable structures in a particular instance of the underlying data model. Typically in ***RDM*** we consider a class of statements in a first-order language, and view these statements as axioms which constrain the set of possible models for the system. Constraints, therefore, are meaningful *only* in a system wherein (a) they can be **expressed**, and (b) they can be **enforced**. It is therefore often convenient to think of the constraints as a part of the schema itself, and consider as valid only those relations which satisfy the domain constraint, the key constraint, and any additional constraints specified.

Historically ***RDM*** was defined before most of the theoretical work on relational constraints, and so especially with respect to the constraints it is never quite clear which class is being included in a discussion of "the relational model." Two classes of constraints have been widely studied [Nicolas 78]:

1. Functional Dependencies (FDs) such as X --> Y, which abbreviates
   $\forall t_1 \forall t_2 [t_1(X) = t_2(X) \rightarrow t_1(Y) = t_2(Y)]$
2. Multivalued Dependencies (MVDs) such as X -->--> Y, which abbreviates
   $\forall t_1 \forall t_2 [t_1(X) = t_2(X) \rightarrow [\exists t_3 [t_3(X) = t_1(X) \wedge t_3(Y) = t_2(Y)]]]$

We have already indicated that for the purposes of this paper we are considering the constraints ***C*** of

RDM to consist exactly of the FDs and MVDs.

It is convenient to think of a constraint c as a function which maps any set of n relations into {0,1} according to whether the relations satisfy or violate the constraint, i.e. $c : \mathbf{1NF}^n \rightarrow \{0,1\}$. We can extend this to a set of constraints $C = \{ c_1, c_2, ..., c_m \}$ and define $C : \mathbf{1NF} \rightarrow \{0,1\}$ as $C = \wedge_{i = 1 \text{ to } n} c_i$.

Then, given a set of constraints C on the set of structures **1NF**, and a consistent structural extension ψ_S, a set of constraints C' would be a consistent extension of C iff C' was true on a set of relations $\{\psi_S(r_1), \psi_S(r_2), ..., \psi_S(r_n)\}$ in the structural extension of ***1NF*** when and only when C was true on the set of relations $\{ r_1, r_2, ..., r_n \}$ from which it was derived. We state this more precisely in the following definition.

Definition: Let $\Sigma = \{ c_1, c_2, ..., c_m \}$ be a logically consistent set of constraints from the class of constraints **C** on any set of relations $R = \{ r_1, r_2, ..., r_n \}$ on schemes $\{ R_1, R_2, ..., R_n \}$ in ***RDM***. A class of constraints ***C'*** is a ***consistent constraint extension*** of the class of constraints ***C*** iff for every consistent structural extension of ***RDM*** ψ_S, and every logically consistent set of constraints Σ in ***C*** on R, there exists a logically consistent set of constraints Σ' in ***C'*** on the image of relations R under ψ_S such that $\Sigma(r_1, r_2, ..., r_n) = \Sigma'(\psi_S(r_1), \psi_S(r_2), ..., \psi_S(r_n))$.

3.3.1. Normalization

Since the attributes in relations which are consistent extensions of ***RDM*** can be associated with structured domains, the standard definitions of ***first, second,*** and ***third normal forms*** are no longer applicable to these models. In this section we discuss a generalized notion of normalization for these extended relations.

Using our definition of a consistent structural extension, the first normal form concept can be extended by saying that all relations that are equivalent to a first normal form relation under an extension function ψ_S are in ***first normal form under*** ψ_S (1NF-ψ_S). However, there is a lack of symmetry between these two notions of first normal form; in ***RDM***, all relations are in first normal form, whereas in an extended relational data model only some of the relations may be in 1NF-ψ_S.

In order to achieve a closer analog to the first normal form of ***RDM*** we take a different approach in defining a version of first normal form that is applicable to extended relational data models.

Let N1NF be a consistent extension under ψ_S of ***RDM*** defined with respect to a structured domain $\boldsymbol{D}$. An extended relation r defined over the relation scheme $\mathbf{R} = \{\mathbf{A}, \mathbf{K}\}$ is in *first normal form with respect to* N1NF if for each attribute $A_i \in \mathbf{A}$, $dom(A_i) \in \boldsymbol{D}$.

With this definition, an "extended" first normal form is the basis for determining the set of allowable relations in a given (extended) relational data model, and is dependent on the domains over which attributes are defined. The traditional first normal form serves the same role in ***RDM***.

In contrast to first normal form (extended and non-extended) which may be viewed as being structurally based, *second normal form* (2NF) and *third normal form* (3NF) are "semantically" based. That is, these two more restricted forms of relations are based on FDs, which in effect are *relationships* perceived by the database designer to exist among a set of attributes.

Since FDs are not based on the type of a domain value, and thus are not constrained by these types, they are applicable, and have the same meaning in any consistent extension to the relational data model. Since both 2NF and 3NF are defined only in terms of FDs they are also applicable to any consistent extension to the relational data model.

3.4. Consistent Extensions

With these preliminary definitions of the structural, operational, and constraint extensions, we can now state the major definition of what constitutes a consistent extension of the relational model.

Definition: A data model <S,θ,C'> is a *consistent extension* of the relational data model **<1NF,RA,C>** if

1. S is a consistent structural extension of **1NF**
2. θ is a consistent operator extension of **RA**

3. C' is a consistent constraint extension of **C**

4. Summary

The notion of a ***consistent extension*** to the relational data model, derived from a homomorphism from the class of ***1NF*** relations to some more general set of relation structures, is proposed. We examine in turn what is meant in this sense to "extend" each of the components of ***RDM***: its structures (relations), its operations (relational algebra) and a family of constraints (here FDs and MVDs). We believe that any proposed extension to ***RDM*** should consistently extend each of these components if it is truly to be considered an "extension." As such the notion of a *consistent extension* provides a framework for defining any extended relational models.

References

•)

[Chen 76] Chen, P.P.S.
The Entity-Relationship Model: Towards a Unified View of Data.
ACM Trans. on Database Systems 1(1):9-36, March, 1976.

[CliffordCroker 87]
Clifford, J., and Croker, A.
The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans,
In *Proc. Third International Conference on Data Engineering.* Los Angeles, February, 1987.

[Codd 70] Codd, E.F.
A Relational Model of Data For Large Shared Data Banks.
Comm. of the ACM 13(6):377-387, June, 1970.

[Codd 81] Codd, E.F.
Data Models in Database Management.
ACM SIGMOD Record 11(2):112-114, February, 1981.

[Codd 82] Codd, E.F.
Relational Database: A Practical Foundation for Productivity.
Comm. of the ACM 25(2):109-117, February, 1982.

[Date 86] Date, C.J.
An Introduction to Database Systems, 4th Ed.
Eddison-Wesley Pub. Co., New York, 1986.

[JaeschkeSchek 82]
Jaeschke, G. and Schek, H.J.
Remarks on the Algebra of Non-First Normal Form Relations.
In *Proceedings of the 1st ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 124-138. 1982.

[Katz 86] Katz, R. H.
Computer-Aided Design Databases.
In Ariav, G. and Clifford, J. (editors), *New Directions for Database Systems*, pages 110--123. Ablex Publishing Co., Norrwood, New Jersey, 1986.

[Maier 83] Maier, D.
The Theory of Relational Databases.
Computer Science Press, Rockville, MD, 1983.

[Nicolas 78] Nicolas, J.M.
First Order Logic Formalization for Functional, Multivalued, and Mutual Dependencies.
In *Proc. ACM SIGMOD 1978*, pages 40-46. Austin, TX, 1978.

[OsbornHeaven 86]
Osborn, S.L. and Heaven, T.E.
The Design of a Relational Database System with Abstract Data Types for Domains.
ACM Trans. on Database Systems 11(3):357-373, September, 1986.

[StonebrakerRowe 86]
Stonebraker, M. and Rowe, L.A.
The Design of POSTGRES.
In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 340-355. Washington, D.C., May, 1986.