Ersch. in: ICDT '86 : Proceedings of the International Conference on Database Theory, Rome, Italy, September 8 - 10, 1986. / Ed. by Giorgio Ausiello and Paolo Atzeni. - Berlin [u.a.] : Springer, 1986. - S. 380-396. - (Lecture notes in computer science ; 243). - ISBN 3-540-17187-8

http://dx.doi.org/10.1007/3-540-17187-8_48

Theoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations

Marc H. Scholl

Technical University of Darmstadt
Department of Computer Science
Alexanderstrasse 24
D-6100 Darmstadt, West Germany

Abstract—Unnormalized (NF^2) relations, not conforming to the first normal form condition (1NF) of the relational model have been proposed recently for a variety of new applications. In this paper we extend NF^2 relational theory in a way that it becomes possible to use NF^2 relations as storage structures for conventional 1NF relational database interfaces. Physical clustering of precomputed joins can be achieved this way without introducing redundancy. However, applying transformation rules to internal relations straightforwardly, will still yield unnecessary join operations. With the equivalence rules introduced here we prove that efficient algebraic optimization can be performed using standard (1NF) techniques. Particularly, all redundant joins can be properly removed.

1. Introduction

A lot of recent research in relational databases has concentrated on unnormalized relations. Dropping the first normal form condition and allowing non-first-normal-form relations (NF^2 relations) has been recognized as a promising attempt to capture the requirements introduced by new ("non-standard") applications of database systems. The areas of interest include textual [SP82], pictorial [BR84], geographical [MSW86], and office data [SLTC82] as well as knowledge representation techniques [SR86]. While experience with the feasibility of the model in these applications is not yet available, the theory of NF^2 relations is already in an advanced state. Formal definitions of the NF^2 relational model have been given by [AB84, FT83, Ja85, RKS85a, SS84] and most of these include operations on NF^2 relations in algebra and/or calculus style. Also design theory for NF^2 relations has already come up with results [AB84, FG85, FT83, Gu85, OY85, RKS85a].

The focus of this paper is different from the above work. As already mentioned in [AB84, FT83, SS83] we consider NF^2 relations to be a reasonable storage structure even for 1NF databases. The underlying idea is the following: the hierarchical structure of NF^2 relations can be utilized as an efficient possibility to store precomputed joins without introducing redundancy. "Denormalization" as a means of internally materializing the most frequent join operations was proposed in [SS80, SS81]. A physical database layout differing from the usual "1 relation—1 operating system file" choice was proven to dramatically reduce query processing costs. The most important but also expensive operation in relational systems is the join. Hence, significant performance enhancements can be achieved with this approach. Redundant storage of tuples from one relation with each matching tuple of the other relation can be avoided, if we use a hierachical storage scheme. Therefore, NF^2 relations were proposed in [SS83] as reasonable storage structures for 1NF databases. Besides the formal description of the internal database layout, as an additional benefit the transformations between conceptual (user) and internal (system) levels, C and I respectively, can be described within the model by the NF^2 relational algebra.

Three related problems with this approach are discussed and solved in this paper. The first one intuitively concerns the information contents of the conceptual and internal levels, which obviously must be identical. Formally, when the transformations are (algebraic) mappings, these must be *invertible*. Invertibility of relational algebra expressions has partly been studied in the past. "Lossless joins" [ASU79] are an example. We will use joins and nesting for the mapping $C \rightarrow I$ to obtain the hierarchical structures. In order to guarantee invertibility of this mapping, we use the outer join rather than natural join to avoid loosing "dangling" tuples. As a consequence of this, we have to deal with a special kind of *null value* introduced by the outer join. It should be clear, that these null values are a formal description of some "dummy" storage objects at an internal DBS layer. Therefore, this paper does not deal with null values in a general context. The interested reader is refered to [RKS85b] for a detailed discussion on several types of nulls in the NF^2 relational model. For our purposes only a few properties of the null value "ω" resulting from the outer join are discussed and the necessary formalism to manipulate these "dummy objects" is defined.

The second and more substantial problem is imposed by the need for algebraic optimization of queries. The algebraic expressions defining the mapping $I \rightarrow C$ can be substituted for the conceptual relations used in the expressions of the user queries. However, algebraic equivalences have to be applied in order to eliminate redundant joins, for instance. Consider two C-relations that are internally stored in one materialized join relation. The C-relations can be reobtained by projections. Thus substituting the projection expressions into user queries yields a join of two projections. Algebraic optimization techniques can be used to recognize the redundant join—according to [ASU79]—and eliminate it from the query. Without this optimization, the efficient internal structure could not be utilized.

Equivalence rules known from previous research [ASU79, Ul82, Ma83] for the 1NF case and also including NF^2 relations [AB84, FT83, JS82] are not sufficient to do the necessary transformations. Extended equivalences are presented in this paper. According to these, all materialized joins in user queries can be properly eliminated.

The third aspect captured by this paper, concerns the NF^2 queries resulting from the optimization step. Obviously, for our hierarchical internal DB layout there is a class of NF^2 queries that reflects this structure and therefore can be evaluated very efficiently. Our objective is to map a maximum of user queries to that kind of internal counterparts. In fact, we will see that select-project-join queries can be transformed to these "single pass processible" NF^2 queries, if all joins are internally materialized. It is exactly this class of NF^2 queries that will be available at the kernel interface of a prototype NF^2 relational database system developed at the University of Darmstadt [DOPSSW85, DPS86, SW86, Sche86].

As an introductory example consider two conceptual (4NF) relations

$$Dept(\underline{dno}, dname, budget)$$
$$Emp(\underline{eno}, ename, sal, dno)$$

If we internally store the NF^2 relation

$$Idept(\underline{dno}, dname, budget, Iemp(\underline{eno}, ename, sal))$$

we do not need to compute $Dept \bowtie Emp$ at runtime of user queries. For instance consider the user query $\pi_{dname,sal}\sigma_{ename='Smith'}(Dept \bowtie Emp)$ which can be processed very efficiently by the

following NF^2 algebraic expression (see section 2 for details).

$$\mu_{Iemp}(\pi[dname, \pi[sal](\sigma_{ename='Smith'}(Iemp))](Idept))$$

The internal query contains only a simple NF^2 filter expression and unnesting but no join operation, thus it can be evaluated efficiently.

Section 2 shortly reviews the NF^2 relational model from [SS84] and introduces the notations used throughout the paper. The class of *single pass processible* queries is informally characterized and a subset of the algebra is claimed to be single pass processible. In section 3 we define the type of conceptual-to-internal mappings foreseen in this paper and deal with the problem of inverting the mapping. We present the necessary rules to deal with the special type of null value resulting from the outer joins.

Extended algebraic equivalence rules involving the null values as well as the nested algebra operators from [SS84] are presented in section 4. With these rules we prove that algebraic optimization can be done in our environment using essentially 1NF optimization and some simple transformation rules. Optimal "quasi 1NF" expressions are computed and then transformed to NF^2 expressions by an optimizer sketched in section 5. The paper concludes with an outline of further research in section 6.

2. The NF^2 Relational Model

Relations with relation valued attributes do not conform to the first-normal-form, because they allow decomposable attribute values, namely relations. Thus, formal definitions like [FT83, RKS85a, SS84] apply 1NF notations repeatedly to capture the nested relational structures. We do not give formal definitions in this paper, but shortly review the notations drawn from [SS84]. As in the 1NF case (cf. [Ma83]) we have a relation R as a pair $\langle d, v\rangle$ consisting of a description d and a value v. The description d is a pair $\langle n, s\rangle$, where n is the name of the relation drawn from some given set N of names. The schema s describes the components of R. Hence s is a set of descriptions d_i of the attributes of R. Atomic attributes have descriptions $\langle a_i, \emptyset\rangle$, i.e. the empty set serves as the schema of atomic attributes. Relation valued attributes have descriptions $\langle a_i, s_i\rangle$, where $s_i \neq \emptyset$ is the schema of the corresponding (sub-) relation. For a relation $R = \langle d, v\rangle$ with $d = \langle n, s\rangle$ we apply the following notation:

$\text{sch}(R) = \text{sch}(d) = s$	the schema
$\text{val}(R) = v$	the value and
$\text{attr}(R) = \{a_i \mid \langle a_i, s_i\rangle \in d\}$	the attributes (name components of $\text{sch}(R)$) of R.

The value of a relation $\text{val}(R)$ is a set of tuples, each tuple can either be regarded as a mapping from an attribute set to a corresponding domain set or as an element of a Cartesian product of domains [SS84, Ja86]. We adopt the first interpretation here and use $t(a_i)$ to denote the a_i-value of a tuple t in $\text{val}(R)$ where $a_i \in \text{attr}(R)$. Extending this notion canonically $t(A)$ is the A-value of t in $\text{val}(R)$ for $A \subseteq \text{attr}(R)$. As usual we give the schema of an NF^2 relation in the following form:

$$R(a_1, \ldots, a_n, b_1(\ldots), \ldots, b_m(\ldots))$$

where the a_i are the names of atomic attributes of R and b_i the names of relation valued attributes. Between the parentheses after b_i we denote the schema of the b_i in exactly the same way.

The NF^2 relational algebra we use in the following has been presented in [SS83] and formally defined in [SS84]. Operators $\times, \cup, \cap, -, \bowtie$ are defined exactly as in the 1NF case, whereas π and σ (projection and selection) have been extended and two new operators ν and μ (nest and unnest) were introduced. The latter two are also used in [AB84, JS82, FT83, RKS85a, SS83, SS84]. Extended defintions for μ and ν will be given in section 3 of this paper, the extensions of π and σ shall be exemplified in the following.

For the scope of this paper, concerning extended selection we only need set comparisons allowed in selection formulae together with set-valued constants (especially $\emptyset$). Within projection lists, i.e. the set of attributes that are projected, we allow the application of algebraic expressions (especially again π and σ) to relation valued attributes. As an example consider a relation

$$Dept(dno, dname, Emp(eno, ename, sal, Course(cno, year)))$$

(with the obvious interpretation) then

$$Q_1 = \sigma[dname = \text{'research'} \wedge Emp \neq \emptyset](Dept)$$

yields 'established' research departments, i.e. those who already have employees,

$$Q_2 = \pi[dname, \pi[ename, sal](Emp)](Dept)$$

gives a list of department names each with a list of employee names and salaries, i.e. the schema of Q_2 is $Q_2(dname, Emp(ename, sal))$ and

$$Q_3 = \pi[dname, \sigma[sal > 10.000](Emp)](Dept)$$

yields a relation with schema $Q_2(dname, Emp(eno, ename, sal, Course(\ldots)))$—a list of "rich" employees grouped by *dname*. Details of the nested algebra are not important in this paper, the interested reader is referred to [SS84]. Nevertheless, we will shortly discuss our notion of ***single pass processible*** queries. By the term single pass processible we indicate, that this kind of queries can be evaluated efficiently by an NF^2 relational DBMS [DPS86, Sche85]. The results of these queries can be computed within one (hierarchical) scan over the tuples in val(R). The notion was influenced by so-called single-table queries in flat relational systems, i.e. queries that do not include joins. However, in our case of nested relations "single table" queries may well contain operations with join complexity (set comparisons, nested joins). Therefore we need a more sophisticated notion of "simplicity" of queries. A detailed discussion of what is single pass processible and what is not, is deferred to a forthcomming report. However, the following theorem describes a class of queries that surely belongs to this class.

THEOREM 2.1. *An NF^2 relational expression is* **single pass processible**, *if it is built according to the following recursive rule:*

$$\text{spe}(R) = \pi[L]\sigma[F]\begin{cases} R \\ \pi[L^*](R) \end{cases}$$

L is a "1NF" projection list, i.e. without nested algebraic expressions

F is a simple selection formula, consisting of conjunctions of comparisons between attributes and constants ($\emptyset$ is the only constant permitted for relation valued attributes) and

L^* is a projection list containing single pass expressions applied to relation valued attributes, i.e. spe(A) ■

Like most results of this paper, due to limitations on space the above is stated without proof.

3. Transformation Between Conceptual and Internal Layer

In this section we will define conceptual schemata, the various choices of conceptual-to-internal mappings and study invertibility of these mappings. Extensions of nest and unnest operators—compared to [AB84, FT83, RKS85a, SS84]—are introduced to capture the null values resulting from outer joins. The latter are needed to guarantee invertibility of the denormalization, because natural joins would loose "dangling" tuples in general.

3.1 Conceptual Relations

We consider a given set "conceptual DB schema" $C = \{CR_1, \ldots, CR_n\}$ of (conceptual) relations CR_i. The schema C is in 4NF [Fa77], which means every valid MVD (and thus FD) is expressed as a key condition in one of the CR_i. Furthermore, we assume C to be free of null values. We denote with $\text{key}(CR_i) \subseteq \text{attr}(CR_i)$ the set of primary key attributes of CR_i. Without loss of generality we assume all attribute names in C distinct except foreign keys. $A \subseteq \text{attr}(CR_i)$ is a foreign key, iff $A = \text{key}(CR_j)$ for some $j \neq i$ and $CR_i, CR_j \in C$. We will sometimes use C_i as a short form for $\text{attr}(CR_i)$. For the following discussion of mappings it is useful to illustrate the conceptual schema by a directed (acyclic) graph.

DEFINITION 3.1. *Let C be a set of relations with known keys and foreign keys. The corresponding* **schema graph** *is $\text{sg}(C) = (N, E)$, where the set of nodes N contains one node per conceptual relation. $N = \{\overline{CR_i} \mid CR_i \in C\}$ and the set of edges E is obtained from the key-foreign key relationships in C: $E = \{\overline{CR_i} \rightarrow \overline{CR_j} \mid \text{key}(CR_i) \subseteq \text{attr}(CR_j) \wedge i \neq j\}$*

A similar schema graph with attribute nodes added is used in [FT83]. $\text{sg}(C)$ corresponds to the Bachman-diagram of the equivalent CODASYL database. In analogy to this, if $\overline{CR_i} \rightarrow \overline{CR_j}$ is in $\text{sg}(C)$ we will call CR_i the "owner relation" of the "member" CR_j in the following.

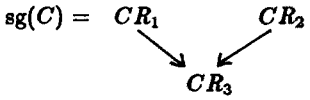
3.2 Conceptual to Internal Mappings

As claimed in the introduction and illustrated by an example our overall goal in choosing internal representations is improving performance. Two aspects of physical database design will be considered in this section: clustering and denormalization. Clustering means mapping data that is accessed together very frequently to the same block or at least adjacent blocks of secondary storage media thus reducing the costs of physical I/O. Denormalization was proposed in [SS80, SS81]. Whenever two relations are combined by (natural) joins in a considerable portion of user queries, it may be advantageous to internally precompute the join and store the result. Obviously, the decision for or against this "materialization" of the join depends on update frequencies for the involved relations. Whereas [SS80, SS81] decide to store the original relations in addition to the join sometimes, we can drop them if the join is materialized. The reason is twofold. First we use the outer join rather than a natural join to avoid loosing tuples and secondly we nest the "member" relation into the "owner" yielding a hierarchical structure. Therefore we do not introduce redundant storage of owner attributes with every matching member as in [SS80, SS81]. Moreover, nesting the members into the owner tuples yields a clustered storage structure. By the combination of join and nest we can establish both physical design objectives mentioned above. Considering the conceptual schema graph $\text{sg}(C)$ we can describe the possible internal NF^2 representations (that do not introduce redundancy) by the following definition.

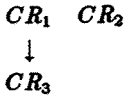
DEFINITION 3.2. *Given a conceptual schema C with its graph* $\mathrm{sg}(C)$, *the set* $\mathrm{ir}(C)$ *of all possible* **internal representations** *obtained by denormalization and nesting is described by the set* $\mathrm{for}(\mathrm{sg}(C))$ *of all forests that can be obtained by deleting some of the edges in* $\mathrm{sg}(C)$.

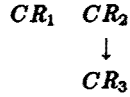
The set of internal NF^2 relations corresponding to $I \in \mathrm{ir}(C)$ is intuitively clear. Every tree in $\mathrm{for}(I)$ is a relation, where the root denotes the top level relation and descendents denote subrelations. In general $\mathrm{ir}(C)$ will contain more than one alternative. It is the task of physical database design (optimization) to select one of them based upon an estimated transaction mix. An example is given in figure 3.1.

$$C = \{CR_1(\underline{A}, B, C)\quad CR_2(\underline{D}, E, F)\quad CR_3(\underline{A, D}, G)\}$$

$\mathrm{sg}(C) =$ 

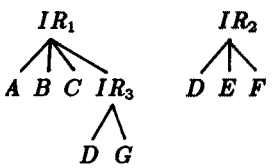
$$\mathrm{ir}(C) = \{I_1, I_2, I_3\}$$

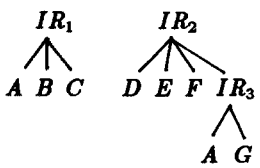
$\mathrm{for}(I_1) =$ 

$\mathrm{for}(I_2) =$ 

$\mathrm{for}(I_3) = CR_1\ CR_2\ CR_3$

corresponding internal relations:

$I_1 = \{IR_1, IR_2\}$ 

$I_2 = \{IR_1, IR_2\}$ 

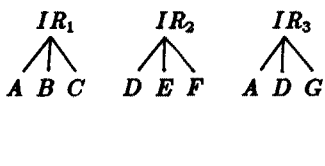
$I_3 = \{IR_1, IR_2, IR_3\}$ 

Figure 3.1: Choices for internal representations

DEFINITION 3.3. *The* **tree representation** *[SS84] of the internal NF^2 relations corresponding to* $\mathrm{for}(I)$ *is obtained by appending nodes* $\overline{a_i}$ *for* $a_i \in attr(CR_i)$ *to the nodes* $\overline{CR_i}$, *iff* $a_i \notin key(CR_j)$ *and* $\overline{CR_j} \rightarrow \overline{CR_i}$ *in* $\mathrm{for}(I)$. *Finally* $\overline{CR_i}$ *is renamed to* $\overline{IR_i}$. *The internal schema I is the set of relations denoted by root nodes in* $\mathrm{for}(I)$.

Every edge in $\mathrm{for}(I)$ corresponds to an outer join between the owner and member relation followed by a nest operation. All attributes of the member except the foreign key are nested into a subrelation.

EXAMPLE 3.1.

$\mathrm{for}(I) =$ 

$\Longrightarrow$

$$I = \{IR_1\}$$

$$IR_1 = \nu_{IR_2=(C_2 \setminus C_1)}(CR_1 \boxtimes CR_2)$$

As we used the outer join operator ($\boxtimes$) to produce null values for dangling tuples, we have to decide on how nest should behave on tuples containing nulls. One decision is obvious: if all attributes to be nested have a null value, then the resulting tuple should have an empty subrelation. However, a problem arises, if all attributes that are *not* nested have a null value. Especially, should several of such occurrences be regarded equal or not. In a more general context, this would lead to a discussion on equality of null values or the concept of "informativeness" of tuples containing nulls—see [RKS85b], for instance. Again we remind the reader, that this paper is not meant as an overall treatment of null values. As we introduced the null values as a formal tool of modelling "dummy objects", we can decide on equality from a pragmatic point of view. For our purpose of defining an internal schema it is appropriate to consider nulls equal in non-nesting attributes, i.e. all member tuples without owners are gathered in a subrelation of a single tuple containing only nulls in the owner attributes. Notice that this situation was excluded in our definition of conceptual schemata, because we did not allow nulls in C-relations. Nevertheless, we give a definition of ν that captures this case too. For instance, consider figure 3.2. Notice that all definitions that follow will not contain the *schema* part of the operations defined. Most of the definitions from [SS84] can be applied. In the other cases, the schema transforming effect of the operations should be obvious. A formal treatment would require additional notational conventions that were left out. The same applies for the proofs of equivalent algebraic expressions.

R			
A	B	C	D
a_1	b_1	ω	ω
a_2	b_2	c_1	d_1
a_2	b_2	c_2	d_2
ω	ω	c_3	d_3
ω	ω	c_4	d_3

$\Longrightarrow$

$\nu_{CD=(C,D)}(R)$			
A	B	CD	
		C	D
a_1	b_1	$\emptyset$	
a_2	b_2	c_1	d_1
		c_2	d_2
ω	ω	c_3	d_3
		c_4	d_3

Figure 3.2: Nesting tuples containing nulls

Here "ω" denotes a null meaning "not existent". (This "closed world assumption" is appropriate, because we want to store existent tuples from the C-relations !)

DEFINTION 3.4. *Let* $\text{attr}(R) = A, B \subseteq A$, *then*

$$\text{val}(\nu_{x=(B)}(R)) = \{t \mid t(A \setminus B) \in \text{val}(\pi_{A \setminus B} R) \wedge t(x) = \{t'(B) \mid t' \in \text{val}(R) \wedge t'(A \setminus B) = t(A \setminus B) \wedge \forall b \in B : t'(b) \neq \omega\}\}$$

Notice how the additional condition $t'(b) \neq \omega$ for all $b \in B$ produces the desired empty subrelations.

3.3 Inversion of the Mapping $C \rightarrow I$

Conversely, unnest should map empty subrelations to nulls rather than eliminating these tuples, as it does in the standard definitions [AB84, FT83, RKS85a, SS84], hence we introduce an unnest operator $\overline{\mu}$ that preserves tuples containing empty subrelations by mapping $\emptyset$ to ω.

DEFINITION 3.5. *Let* $\text{attr}(R) = A$, $x \in A$ *and* $\text{attr}(x) = B$, *then*

$$\text{val}(\overline{\mu}_{x=(B)}(R)) = \{t \mid \exists t' \in \text{val}(R) : t(A \setminus B) = t'(A \setminus B) \wedge \\ \big((t(B) \in t'(x) \wedge t'(x) \neq \emptyset) \vee (\forall b \in B : t(b) = \omega \wedge t'(x) = \emptyset) \big) \}$$

The conceptual relations did not contain null values, hence we need an operation, to eliminate tuples containing nulls. Therefore, we introduce a reduction operator ρ_X, deleting all tuples from a relation, containing a null value in at least one of the attributes in X:

DEFINITION 3.6. *Let* R *be a relation with* $X \subseteq \text{attr}(R)$, *then*

$$\text{val}(\rho_X(R)) = \{t \mid t \in \text{val}(R) \wedge \forall a \in X : t(a) \neq w\}$$
$$\rho(R) = \rho_{\text{attr}(R)}(R)$$

Obviously, whether we unnest first and eliminate ω-tuples afterwards or delete tuples containing empty sets and then unnest yields the same result. Further, the standard unnest operator μ comprises ρ and $\overline{\mu}$.

LEMMA 3.1.

$$\rho_X \overline{\mu}_{C=(X)}(R) = \overline{\mu}_{C=(X)} \sigma_{C \neq \emptyset}(R) = \mu_{C=(X)}(R)$$ ■

It is known from [FT83, JS82] that any nest operation can be undone by applying the corresponding unnest, i.e. $\mu_A \nu_A(R) = R$. Our simple extension of the nest operation, however, invalidates this property in general. Consider two tuples $t_1 = \langle a, \omega \rangle$ and $t_2 = \langle a, b \rangle$; nesting the second attribute produces $t' = \langle a, \{b\} \rangle$. A subsequent unnest will not reproduce tuple t_1. The reason is that there was another tuple (t_2) containing the same information in the attributes not nested, and thus nest did not map t_1 to $\langle a, \emptyset \rangle$ (t_2 is *more informative* than t_1). We could think about a more sophisticated definition of nest, but fortunately for our purpose the current definition is sufficient, because the above situation does not occur as we see from the following lemmas.

LEMMA 3.2. *Iff* $\forall t \in \text{val}(R) : \big(t(A) = \omega \Rightarrow \nexists t' \in \text{val}(R) : t'(\text{attr}(R) \setminus A) = t(\text{attr}(R) \setminus A)\big)$ *then*

$$\overline{\mu}_A \nu_A(R) = R$$ ■

LEMMA 3.3. *Let* $t \in \text{val}(R \boxtimes S)$ *and* $a, a' \in \text{attr}(R) \setminus \text{attr}(S)$, *then*

$$t(a) = \omega \Longleftrightarrow t(a') = \omega$$ ■

COROLLARY 3.1.

$$R \boxtimes S = \overline{\mu}_X \nu_{X=(\text{attr}(R) \setminus \text{attr}(S))} R \boxtimes S$$ ■

This means, nesting member tuples as a subrelation into the corresponding owner tuples is always invertible. The following lemma captures the well known fact that outer join is always invertible:

LEMMA 3.4.

$$R = \pi_{\text{attr}(R)}\, \rho_{\text{attr}(R)}\, (R \boxtimes S)$$ ■

With lemma 3.3 we further conclude that instead of checking all attributes of R against null values it is sufficient to check either one except the join attributes:

COROLLARY 3.2.

$$\forall a \in \text{attr}(R) \setminus \text{attr}(S) : \; R = \pi_{\text{attr}(R)} \rho_a (R \boxtimes S)$$ ■

For our example 3.1 we can therefore define the inversion of the mapping $C \rightarrow I$ by:

EXAMPLE 3.1 (CONT'D).

$$IR_1 = \nu_{IR_2=(C_2 \setminus C_1)}(CR_1 \boxtimes CR_2) \implies \begin{array}{l} CR_1 = \pi_{C_1} \rho_{C_1}(IR_1) \\ CR_2 = \pi_{C_2} \rho_{C_2} \overline{\mu}_{IR_2}(IR_2) = \pi_{C_2} \mu_{IR_2}(IR_1) \end{array}$$

Notice that the simplification $CR_1 = \pi_{C_1}(IR_1)$ is valid, if the foreign key condition is maintained on level C, because in this case no "member" tuples would be accepted with a foreign key pointing to a nonexistent "owner". As we did not allow null values in C-relations, there is also no member without an owner. Further we see from example 3.1 that μ-operations are sufficient in the mapping $I \rightarrow C$, if we only unnest those subrelations of IR_i needed to reconstruct a certain CR_j.

Our basic idea for the optimization step described in sections 4 and 5 is the following: use a global unnest-operation, i.e. one that procudes the equivalent 1NF relation (e.g. UNNEST* in [FT83] = μ^* in [RKS85a]) in all of the equations for the mapping $I \rightarrow C$. This yields the same 1NF relation $\mu^*(IR_i)$ in all equations for conceptual relations that are contained in a common IR_j. The benefit is, that to eliminate redundant joins from queries involving such conceptual relations, we could then utilize 1NF techniques. The following consideration explains, why we introduced $\overline{\mu}$ and will use $\overline{\mu}^*$ in the sequel. Consider an owner relation CR_1 with two members CR_2, CR_3. A tuple in IR_1 (with both joins materialized, i.e. subrelations IR_2, IR_3) that has members in IR_2 but not in IR_3 would disappear, if we used μ^*. Hence, we use $\overline{\mu}^*$ preserving all tuples and decide on nulls afterwards using the ρ-operation.

In order to prepare for the next theorem we give a lemma, that allows to bring the equations defining $C \rightarrow I$ into a canonical form.

LEMMA 3.5. *Let* $X \subseteq \text{attr}(R) \setminus \text{attr}(S)$, *then*

$$\nu_{a=(X)}(R \boxtimes S) = \nu_{a=(X)} R \boxtimes S$$ ■

Any internal relation is obtained by a sequence of "outer join followed by nest"-operations, i.e.

$$IR_j = \nu_{IR_{j_1}=(CR_{j_1} \setminus CR_{j_2})}(CR_{j_2} \boxtimes (\nu_{\ldots}(\cdots \boxtimes \cdots)))$$

According to lemma 3.5 this can be rewritten as $IR_j = \nu_{IR_{j_1}=(CR_{j_1} \setminus CR_{j_2})}(\nu_{\ldots}(\nu \cdots (CR_{j_1} \boxtimes (\cdots \boxtimes \cdots (\cdots)))))$, i.e. a sequence of outer joins followed by a sequence of nest operations. Now it is easy to prove theorem 3.1, capturing the invertibility proposition of our conceptual to internal mapping. The inversion can be done by equations in a canonical form using global unnest:

THEOREM 3.1. *Given $C = \{CR_1, \ldots, CR_n\}$ a set of conceptual relations and I a corresponding internal representation $I \in \mathrm{ir}(C)$. For $1 \leq i \leq n$ let $\overline{IR_{j_i}}$ be the root of the tree in* $\mathrm{for}(I)$ *containing node $\overline{IR_i}$, then:*

$$CR_i = \pi_{C_i} \rho_{C_i} \overline{\mu}^*(IR_{j_i})$$ ■

4. Algebraic Equivalences

For the classical relational model a large scale of algebraic equivalences have been studied, a summary of the results is contained e.g. in [Ul82]. [ABU79, ASU79] have given lossless join properties that can be used to eliminate redundant joins from queries expressed in 1NF algebra or the like. Concerning the NF^2 relational model [JS82] presented some first equivalence rules capturing inversion of nest by unnest and vice versa, commutativity of nest and unnest and a few other properties. The work of [FT83] gives a comprehensive overview of positive and negative results on commuting nest/unnest with the other algebraic operators. In case of an extended algebra for NF^2 relations like [AB84, SS84] first results have also been presented. Obviously, with an extended algebra we can prove some more properties than with a standard algebra enriched by nest and unnest only. Here we summarize the main equivalence rules that can be used to efficiently optimize and process select-project-join (SPJ-) queries on level C by NF^2 queries on level I.

As ρ-operations are contained in the expressions we need additional transformation rules for commuting ρ and the other algebraic operators. These are given in subsequent lemmas. We also need criteria for deciding whether the join of two projections equals the original relation (lossless join property) when reduction operations are involved. This can be regarded as an extension of the results from [ABU79]. Theorem 4.1 states the corresponding proposition. Using these results, we can further prove that neglecting the reductions ρ during the join elimination phase and inserting them in the right places afterwards yields correct results.

In the following we will use $X, Y, \ldots$ to denote sets of attributes and adopt the usual short notation XY for the union $X \cup Y$. First we state properties on commuting ρ with other algebra operations, notice that ρ is some special kind of selection.

LEMMA 4.1.

(1) $$\rho_X \rho_Y R = \rho_{XY} R$$

(2) $$\rho_X \sigma_F R = \sigma_F \rho_X R$$

(3) $$\rho_X \pi_Y R = \pi_Y \rho_X R \qquad \Longleftrightarrow X \subseteq Y$$

(4) $$\rho_{XY}(R \times S) = \rho_X R \times \rho_Y S \iff X \subseteq \mathrm{attr}(R) \wedge Y \subseteq \mathrm{attr}(S)$$

(5) $$\rho_{XY}(R \bowtie S) = \rho_X R \bowtie \rho_Y S \iff X \subseteq \mathrm{attr}(R) \wedge Y \subseteq \mathrm{attr}(S)$$

(6) $$\nu_{a=(X)} \rho_X R = \sigma_{a \neq \emptyset} \nu_{a=(X)} R$$

(7) $$\rho_X \overline{\mu}_{a=(X)} R = \overline{\mu}_{a=(X)} \sigma_{a \neq \emptyset} R = \mu_{a=(X)} R$$ ■

LEMMA 4.2.

$$\pi_{XY}(R \bowtie S) = \pi_X R \bowtie \pi_Y S \iff X \cap Y = \mathrm{attr}(R) \cap \mathrm{attr}(S)$$

Proof: Substitute π, σ and $\times$ for $\bowtie$ and apply the equivalences known from [Ul82]. ■

Combining the two lemmas we derive the following theorem giving a lossless join criterion for project-join mappings containing ρ's:

THEOREM 4.1. *Let* $X, X', Y, Y' \subseteq \text{attr}(R)$, *then*

$$\pi_{XY}\rho_{Y}R \bowtie \pi_{X'Y'}\rho_{Y'}R = \pi_{XX'Y}\rho_{YY'}R$$

$$\Longleftrightarrow$$

$$X \cap X' = XY \cap X'Y' \wedge \left(X \cap X' \twoheadrightarrow XY \vee X \cap X' \twoheadrightarrow X'Y'\right)$$

Proof:

$$\begin{aligned}
\pi_{XY}\rho_{Y}R \bowtie \pi_{X'Y'}\rho_{Y'}R &= \pi_{X}(\pi_{XY}\rho_{Y}R) \bowtie \pi_{X'}(\pi_{X'Y'}\rho_{Y'}R) \\
&= \pi_{XX'}(\pi_{XY}\rho_{Y}R \bowtie \pi_{X'Y'}\rho_{Y'}R) \\
&\qquad \Longleftrightarrow X \cap X' = XY \cap X'Y' \text{ (Lemma 4.2)} \\
&= \pi_{XX'}(\rho_{Y}\pi_{XY}R \bowtie \rho_{Y'}\pi_{X'Y'}R) \\
&= \pi_{XX'}\rho_{YY'}(\pi_{XY}R \bowtie \pi_{X'Y'}R) \\
&= \pi_{XX'}\rho_{YY'}\pi_{XYX'Y'}R \\
&\qquad \Longleftrightarrow XY \cap X'Y' \twoheadrightarrow XY \text{ or } X'Y' \text{ ([ABU79])} \\
&= \pi_{XX'}\rho_{YY'}R
\end{aligned}$$

We applied lemma 4.1 in the 3rd, 4th and 6th equation. The preconditions for the 2nd and 5th equations conclude the proof. ■

Provided with the above equivalences we can transform user queries from the conceptual to the internal level by substituting the equations for CR_i (from $I \rightarrow C$) into the C-query. Optimization and join elimination can be performed by applying these results. In this "verbatim" approach, however, we have to deal with reduction operations (ρ). Thus we have to develop a new formula manipulation system, or whatever method else. Our claim in the beginning was, that *traditional 1NF* techniques would suffice. In order to prove this statement we give a variation of the lossless join condition in the following. Due to this new criterion we can rely on join elimination that does not consider ρ's. The only prerequisite is that we introduce the appropriate reductions into the optimized query expressions afterwards.

THEOREM 4.2. *For some conceptual relations* CR_1, CR_2 *internally stored in* IR, *with attribute sets* C_1, C_2 *respectively the following equivalence holds:*

(*i*) $$\pi_{X}(\pi_{C_1}\rho_{C_1}\overline{\mu}^{*}IR) \bowtie \pi_{X'}(\pi_{C_2}\rho_{C_2}\overline{\mu}^{*}IR) = \pi_{XX'}\pi_{C_1C_2}\rho_{C_1C_2}\overline{\mu}^{*}IR$$

$$\Longleftrightarrow$$

(*ii*) $$\pi_{X}\overline{\mu}^{*}IR \bowtie \pi_{X'}\overline{\mu}^{*}IR = \pi_{XX'}\overline{\mu}^{*}IR$$

Proof: (*i*) $\Rightarrow$ (*ii*): Easy (and not used in the following)
(*i*) $\Leftarrow$ (*ii*):

$$(ii) \Rightarrow X \cap X' \twoheadrightarrow X \text{ or } X \cap X' \twoheadrightarrow X' \text{ in } \overline{\mu}^{*}IR$$

The outer join does not introduce any MVDs, except the following:

$$R = R_1 \boxtimes R_2 \Longrightarrow R_1 \cap R_2 \twoheadrightarrow R_1 \text{ and } R_1 \cap R_2 \twoheadrightarrow R_2 \text{ in } R$$

Hence, $X \cap X' \twoheadrightarrow X$ or $X \cap X' \twoheadrightarrow X'$ must be valid MVDs in C. As C was claimed to be in 4NF there are no nontrivial MVDs except key conditions. Thus there must exist CR_1, CR_2 such that $X \subseteq C_1$, $X' \subseteq C_2$, with $X \cap X' = C_1 \cap C_2$ and $X \cap X' \twoheadrightarrow C_1$ or $X \cap X' \twoheadrightarrow C_2$. From this and theorem 4.1 we conclude (*ii*) $\Rightarrow$ (*i*) which completes the proof. ■

5. An NF^2 Algebraic Query Optimizer

In this section we will emphasize the practical issues of our theory. The conclusion will be that in fact 1NF optimization is the essential part needed in our proposed environment. However, before we can outline a prototype implementation we have to state a few more algebraic equivalences involving the nested NF^2 algebra from [SS84].

From theorem 4.2 we conclude that substituting the expression from $I \rightarrow C$ and applying 1NF join elimination plus rules to move selections and projections down the expression tree (which reduces the size of intermediate results and is a standard algebraic optimization technique—see [Ul82]) yields an optimized "quasi-1NF" relational expression of the form

$$E = \pi\sigma(\bowtie(\pi\sigma(\rho\overline{\mu}^* IR_j)))$$

that does not contain redundant joins anymore. By the term "quasi-1NF" we indicate the fact, that the essential part of the expression uses only 1NF algebra. However, due to the presence of $\overline{\mu}^*$ we actually have an NF^2 expression.

The strategy now is to push the unnest operations up in the query's parse tree in order to utilize the power of the NF^2 algebra. In fact we will not apply $\overline{\mu}^*$, but rather the necessary $\overline{\mu}$'s only. As a result we will observe a single type of NF^2 expressions resulting from this process, namely *single pass processible queries* (or joins of those to be exact). For the scope of this paper we will only give the rules to commute $\overline{\mu}$ and the σ and π operations that follow in the above expression E. Some types of joins with subrelations can be performed within the NF^2 algebra, results on this topic will be presented in a forthcomming report. The transformation rules for ρ and π/σ are given in the following theorems, which are stated without proofs, because the formalism needed was left out in this paper.

THEOREM 5.1. *If $F = F_1 \wedge F_2$ where F_1 refers to attributes in* $\mathrm{attr}(R) \setminus \{A\}$ *and F_2 refers to* $\mathrm{attr}(A)$, *then*

$$\sigma[F]\,\overline{\mu}_A\,(R) = \overline{\mu}_A\,\sigma[F_1]\,(\pi[\,\mathrm{attr}(R) \setminus A,\ \sigma[F_2](A)]\,(R))$$ ■

THEOREM 5.2. *If $L = L'L''$ where $L' \subseteq \mathrm{attr}(R) \setminus A$, $L'' \subseteq \mathrm{attr}(A)$, then*

$$\pi[L]\,\overline{\mu}_A\,(R) = \overline{\mu}_A\,\pi[\,L',\ \pi[L''](A)\,]\,(R)$$ ■

We give example applications of these transformation rules in the following.

EXAMPLE 5.1. *Let $R = R(A_1(A_{11}(A_{111}, A_{112})), A_2)$ then*

$$\sigma_{A_{111}=5 \wedge A_2=3}\,\overline{\mu}_{A_{11}}\,(\overline{\mu}_{A_1}(R))$$

$$\downarrow$$

$$\overline{\mu}_{A_{11}}\,\sigma_{A_2=3}\left(\pi[\ldots,\,\sigma_{A_{111}=5}(A_{11})]\,(\overline{\mu}_{A_1}(R))\right)$$

$$\downarrow$$

$$\overline{\mu}_{A_{11}}\,\sigma_{A_2=3}\left(\overline{\mu}_{A_1}\left(\pi\left[A_2,\,\pi[\sigma_{A_{111}=5}(A_{11})](A_1)\right](R)\right)\right)$$

$$\downarrow$$

$$\overline{\mu}_{A_{11}}\,\overline{\mu}_{A_1}\,\sigma_{A_2=3}\pi\left[A_2,\pi[\sigma_{A_{111}=5}(A_{11})](A_1)\right](R)$$

EXAMPLE 5.2. *Here we illustrate the treatment of the reductions ρ. Let* $\mathrm{sg}(C) = \{\overline{CR_1} \rightarrow \overline{CR_2}\}$ *and* $I = \{\, IR_1((\mathrm{attr}(CR_1)), IR_2(\mathrm{attr}(CR_2) \setminus \mathrm{attr}(CR_1)))\,\}$. *This means the join between the "owner" CR_1 and the "member" CR_2 is materialized. Consider the following user query:*

$$Q_C = \pi_{C_1}(CR_1 \bowtie CR_2)$$

Notice that $Q_C \neq CR_1$ in general! Following are the equations defining $C \rightarrow I$ and $I \rightarrow C$:

$$\begin{aligned} C \rightarrow I : IR_1 &= \nu_{IR_2=(C_2 \setminus C_1)}\,(CR_1 \boxtimes CR_2) \\ I \rightarrow C : CR_1 &= \pi_{C_1}\rho_{C_2}\overline{\mu}^* IR_1 \\ CR_2 &= \pi_{C_2}\rho_{C_2}\overline{\mu}^* IR_1 \end{aligned}$$

Knowing $C_1 \cap C_2 \rightarrow C_1$, we conclude from theorem 5.1 that the join is redundant, and hence Q_C can be simplified to $\pi_{C_1}\rho_{C_1C_2}\overline{\mu}^* IR_1$. The following is a derivation of this result by application of the rules from section 4:

$$\begin{aligned} Q_C &= \pi_{C_1}(CR_1 \bowtie CR_2) \\ &= \pi_{C_1}\left(\pi_{C_1}\rho_{C_1}\overline{\mu}^* IR_1 \bowtie \pi_{C_2}\rho_{C_2}\overline{\mu}^* IR_1\right) \\ &= \pi_{C_1}\left(\rho_{C_1}(\pi_{C_1}\overline{\mu}^* IR_1) \bowtie \rho_{C_2}(\pi_{C_2}\overline{\mu}^* IR_1)\right) \\ &= \pi_{C_1}\rho_{C_1C_2}\left(\pi_{C_1}\overline{\mu}^* IR_1 \bowtie \pi_{C_2}\overline{\mu}^* IR_1\right) \\ &= \pi_{C_1}\rho_{C_1C_2}\pi_{C_1C_2}\overline{\mu}^* IR_1 \\ &= \pi_{C_1}\rho_{C_1C_2}\overline{\mu}^* IR_1 \\ &= \pi_{C_1}\rho_{C_1}\overline{\mu}^*\sigma_{C_2\neq\emptyset} IR_1 \\ &= \pi_{C_1}\rho_{C_1}\sigma_{C_2\neq\emptyset} IR_1 \\ &= \pi_{C_1}\sigma_{C_2\neq\emptyset} IR_1 \end{aligned}$$

the last equation holds if foreign key conditions are maintained, the fifth one is valid because of the FD given. Notice that without the ρ-operation we would have concluded $Q_C = \pi_{X_1}\pi_{C_1C_2}\overline{\mu}^* IR_1 = \pi_{C_1}\overline{\mu}^* IR_1 = CR_1$, which is not correct!

EXAMPLE 5.3. *Now we derive the result already presented in the introductory example. The equations defining the mappings between conceptual and internal mappings are:*

$$\begin{aligned} C \rightarrow I : Idept &= \nu_{eno,ename,sal:Iemp}\,(Dept \boxtimes Emp) \\ I \rightarrow C : Dept &= \pi_{dno,dname,budget}\rho_{dno,dname,budget}\overline{\mu}^* Idept \\ Emp &= \pi_{eno,ename,sal,dno}\rho_{eno,ename,sal,dno}\overline{\mu}^* Idept \end{aligned}$$

The transformation of the query can be done as follows:

$$\pi_{dname,sal}\sigma_{ename='\text{Smith}'}(Dept \bowtie Emp)$$

$$= \pi_{dname,sal}\sigma_{ename='\text{Smith}'}$$

$$(\pi_{dno,dname,budget}\rho_{dno,dname,budget}\overline{\mu}^{*}\,Idept \bowtie \pi_{eno,ename,sal,dno}\rho_{eno,ename,sal,dno}\overline{\mu}^{*}\,Idept)$$

$$= \pi_{dname,sal}\sigma_{ename='\text{Smith}'}\rho_{dno,dname,budget,eno,ename,sal}$$

$$(\pi_{dno,dname,budget}\overline{\mu}^{*}\,Idept \bowtie \pi_{eno,ename,sal,dno}\overline{\mu}^{*}\,Idept)$$

$$= \pi_{dname,sal}\sigma_{ename='\text{Smith}'}\rho_{dno,dname,budget,eno,ename,sal}(\pi_{dno,dname,budget,eno,ename,sal}\overline{\mu}^{*}\,Idept)$$

$$= \pi_{dname,sal}\sigma_{ename='\text{Smith}'}\rho_{dno,dname,budget,eno,ename,sal}(\overline{\mu}^{*}\,Idept)$$

$$= \pi_{dname,sal}\sigma_{ename='\text{Smith}'}\rho_{dname,sal}(\overline{\mu}^{*}\,Idept)$$

$$= \rho_{dname,sal}\pi_{dname,sal}\sigma_{ename='\text{Smith}'}(\overline{\mu}^{*}\,Idept)$$

$$= \rho_{dname,sal}\pi_{dname,sal}\overline{\mu}^{*}\pi[dno, dname, budget, \sigma_{ename='\text{Smith}'}(Iemp)](Idept)$$

$$= \rho_{dname,sal}\overline{\mu}^{*}\pi[dname, \pi[sal](Iemp)]\pi[dno, dname, budget, \sigma_{ename='\text{Smith}'}(Iemp)](Idept)$$

$$= \rho_{dname,sal}\overline{\mu}^{*}\pi[dname, \pi[sal]\sigma_{ename='\text{Smith}'}(Iemp)](Idept)$$

$$= \rho_{dname}\overline{\mu}^{*}\sigma_{Iemp\neq\emptyset}\pi[dname, \pi[sal]\sigma_{ename='\text{Smith}'}(Iemp)](Idept)$$

Notice that we did not check against null values in the formulation given in the introduction, i.e. the reduction operation and the selection on Iemp were omitted there.

One of the last transformations in the above example combined two subsequent NF^2 projections into one. The following equivalence states the corresponding extension of the classical cascaded projections rule from [Ul82] for our case of nested projections:

LEMMA 5.1. *Let $expr_1, expr_2$ be algebraic expressions (operator sequences), then*

$$\pi[expr_1(A)]\big(\pi[expr_2(A)](R)\big) = \pi[expr_1(expr_2(A))](R) \qquad \blacksquare$$

Now we are able to state the main result of our work contained in this paper. As claimed in the beginning, using NF^2 relations internally does not introduce additional complexity to the join elimination process and thus to algebraic optimization.

THEOREM 5.3. *Given C in 4NF, I as NF^2 relations obtained by denormalization and nesting according to definitions 3.2, 3.3. Any conjunctive SPJ (select-project-join) query on C can be optimized and transformed to I using only 1NF optimization techniques and additionally the rules in theorems 5.1, 5.2 and lemma 5.5. Whenever all joins contained in the conceptual query are materialized internally, this results in only* **one single pass processible** *NF^2 query.* ■

As the applications of the transformation rules from above are straightforward and can be realized efficiently, the complexity of the whole process is dominated by join elimination (i.e. the 1NF optimization). Hence, with the appropriate restrictions on conceptual queries one can for instance optimize in polynominal time using tableaux techniques [ASU79].

A prototype algebraic optimizer has already been implemented in PROLOG. The optimization algorithm can be sketched as follows:

Step 1 – Analyse the parse tree.
If there are two leaf nodes referring to the same internal relation, consider the projections that occur on the path from the leaves to their lowest common ancestor in the tree and find out, whether the join is redundant (applying knowledge about keys, FD's and MVD's).

Step 2 – Move π and σ as far down the tree as possible.

Step 3 – Consider the $\overline{\mu}^*$-operations: apply the necessary unnests and move $\overline{\mu}$ up the tree.

Implicitely, step 1 applies rules from [Ul82] to move selections up in the tree past joins— in order to check redundancy of the join according to [ASU79]. However, as selections can always be moved *up* (but not down), we do not actually transform the tree but simply ignore selections, when we decide whether a join is redundant. If a join is eliminated, the selections from both subtrees are conjunctively combined. In step 2 we perform the usual optimization, trying to move σ (and π) down the parse tree. Finally, step 3 transforms the optimized "quasi 1NF" query to nested NF^2 algebra. Step 3 is the only one using the rules for NF^2 algebra operations, whereas the others are standard 1NF techniques. In step 3 we also introduce the necessary ρ-operations, or apply the corresponding σ (eliminating empty subrelations before the unnest).

Allthough we used PROLOG in this first implementation and added some more equivalences than the ones presented here—concerning disjunctions in selection formulae—response times of the optimizer are fairly short (An adhoc test in a multi-user environment on a SIEMENS mainframe, optimizing the query from example 5.3—including dialogue handling—took about $1.7sec$ CPU-time and $< 5sec$ response time.) Of course it is known that in general optimization is NP-complete [ASU79]. Nevertheless computation time seems to be tolerable in all practical cases.

6. Conclusion

The use of unnormalized relations as a storage structure for 4NF relations has been proposed. As a consequence, the mapping between the differing conceptual and internal database layouts could be defined algebraically. A series of new equivalence rules among algebraic expressions containing unnest and reduction operators as well as NF^2 selections and projections were presented. Based on these equivalences we proved the applicability of our approach. Algebraic optimization needed for elimination of materialized, hence redundant joins can be performed utilizing only 1NF techniques with little additional information for the case of NF^2 relations. Moreover, whenever queries do not need any join operations internally, because all joins contained in the user query are materialized, a simple, efficiently evaluable type of NF^2 expressions results from the transformation. We also sketched a first implementation having proven practicability of the results presented.

From the above observation we also conclude some justification of the way our algebra was designed. Allthough [AB84] use a different type of algebra for their Verso model, they come to a similar result. [Bi85] also states that select-projection-join queries can be mapped to a single "Verso superselection" under some circumstances. While parts of the Verso model will be implemented in hardware (the Verso DB machine [BRS82, V86]), we are pursuing a software implementation. A database *kernel* system, called DASDBS—**DA**rm**S**tadt **D**ata**B**ase **S**ystem— is currently being implemented at the Technical University of Darmstadt [DOPSSW85, Sche86, SW86]. The architecture is partitioned into an application independent kernel, and several application specific front-ends ("DBS family"). All functions that are exspected to be common to the various application areas planned (conventional 4NF interface, geographic applications, knowledge representation

support) will be implemented in the kernel. Currently the kernel will support NF^2 relations and single pass expressions. The application specific front-end for the conventional (flat) relational interface will be built on top of this kernel. The results presented in this paper are the theoretical foundation of the query optimizer contained in this "standard applications module".

Related work for the future includes treatment of disjunctive selection criteria, utilization of nested NF^2 joins and more sophisticated conceptual to internal mappings reflecting e.g. $n : m$-relationships. With little additional (foreign) key redundancy we could support access in a more symmetrical way than by splitting into disjoint hierarchies. The algebraic manipulations needed to optimize in the presence of the new internal structures can easily be added to our solution.

Acknowledgement

Thanks to Hans-Jörg Schek for initiating my work on this subject. Bernd Paul and Gerhard Weikum have contributed by clarifying discussions on the contents and presentation of this paper. Liz Klinger needed little TEXnical support to do the typing.

7. References

[AB84] S. Abiteboul, N. Bidoit: *Non First Normal Form Relations to Represent Hierarchically Organised Data*, Proc. 3rd ACM PODS, Waterloo, Ontario, Canada, 1984

[ABU79] A.V. Aho, C. Beeri, J.D. Ullman: *The Theory of Joins in Relational Databases*, ACM TODS, Vol. 4:3, 1979

[ASU79] A.V. Aho, Y. Sagiv, J.D. Ullman: *Equivalences Among Relational Expressions*, SIAM Journ. of Computing, Vol. 8:2, 1979

[Bi85] N. Bidoit: *Efficient Evaluation of Queries Using Nested Relations*, Techn. Report INRIA, 1985

[BR84] W. Benn, B. Radig: *Retrieval of Relational Structures for Image Sequence Analysis*, Proc. 10th VLDB Conference, Singapore, 1984

[BRS82] F. Bancilhon, P. Richard, M. Scholl: *On Line Processing of Compacted Relations*, Proc. 8th VLDB Conference, Mexico, 1982

[DOPSSW85] U. Deppisch, V. Obermeit, H.-B. Paul, H.-J. Schek, M.H. Scholl, G. Weikum: *The Storage Component of a Database Kernel System*, Techn. Rep. DVSI–1985–T1, TU Darmstadt,
Short German version in: Proc. GI Conf. on DBSs in Office, Technical and Scientific Applications, Karlsruhe, 1985, IFB 94, Springer, german title: *Ein Subsystem zur stabilen Speicherung versionenbehafteter, hierarchisch strukturierter Tupel*

[DPS86] U. Deppisch, H.-B. Paul, H.-J. Schek: *A Storage System for Complex Objects*, Proc. Int'l Workshop on Object-Oriented Database Systems, Pacific Grove, CA, 1986

[Fa77] R. Fagin: *Mulitvalued Dependencies and a New Normal Form for Relational Databases*, ACM TODS, Vol. 2:3, 1977

[FG85] P.C. Fischer, D. van Gucht: *Determining When a Structure is a Nested Relation*, Proc. 11th VLDB Conference, Stockholm, 1985

[FT83] P.C. Fischer, S.J. Thomas: *Operators for Non-First-Normal-Form Relations*, Proc. IEEE COMPSAC 1983

[Gu85] D. van Gucht: *Theory of Unnormalized Relational Structures*, Ph. D. Thesis, also available as Techn. Rep. CS–85–07, Vanderbilt University, Nashville, TN, 1985

[Ja85] G. Jaeschke: *Recursive Algebra for Relations with Relation Valued Attributes*, Techn. Rep. TR 85.03.002, IBM Heidelberg Scientific Centre, 1985

[Ja86] G. Jaeschke: *Algebraic Expressions for Higher Order Relational Databases*, unpublished manuscript

[JS82] G. Jaeschke, H.-J. Schek: *Remarks on the Algebra of Non-First-Normal-Form Relations*, Proc. 1st ACM PODS, Los Angeles, 1982

[Ma83] D. Maier: *The Theory of Relational Databases*, Pitman Publishing Ltd., London, 1983

[MSW86] F. Maher, H.-J. Schek, W. Waterfeld: *A Database Kernel System for Geoscientific Applications*, to appear in: Proc. 2nd Int'l Symp. on Spatial Data Handling, Seattle, 1986

[OY85] Z.M. Ozsoyoglu, L.Y. Yuan: *A Normal Form for Nested Relations*, Proc. 4th ACM PODS, 1985

[RKS85a] M.A. Roth, H.F. Korth, A. Silberschatz: *Extended Algebra and Calculus for ¬1NF Relational Databases*, Techn. Rep. TR-84-36, Revised Version, University of Texas at Austin, 1985

[RKS85b] M.A. Roth, H.F. Korth, A. Silberschatz: *Null Values in ¬1NF Relational Databases*, Techn. Rep. TR-85-32, University of Texas at Austin, 1985

[Sche85] H.-J. Schek: *Towards a Basic Relational NF^2 Algebra Processor*, Proc. Int'l Conf. on FODO, Kyoto, 1985

[Sche86] H.-J. Schek: *Research Activities of the DVSI-Group 1983-1985*, Techn. Rep. DVSI-1986-T2, TU Darmstadt, 1986

[SLTC82] N.C. Shu, V.Y. Lum, F.C. Tung, C.L. Chang: *Specification of Forms Processing and Business Procedures for Office Automation*, IEEE TOSE, Vol. SE-8:5, 1982

[SP82] H.-J. Schek, P. Pistor: *Data Structures for an Integrated Database Management and Information Retrieval System*, Proc. 8th VLDB Conference, Mexico, 1982

[SR86] H.-J. Schek, U. Reimer: *A Frame Representation Model and its Mapping to NF^2 Relations*, submitted for publication, 1986

[SS80] M. Schkolnik, P. Sorenson: *Denormalization: A Performance Oriented Database Design Technique*, Proc. AICA Conf., Bologna, Italy, 1980

[SS81] M. Schkolnik, P. Sorenson: *The Effects of Denormalization on Database Performance*, Res. Rep. RJ3082 (38128), IBM Res. Lab. San Jose, Ca., 1981

[SS83] H.-J. Schek, M.H. Scholl: *Die NF^2-Relationenalgebra zur einheitlichen Manipulation externer, konzeptueller und interner Datenstrukturen*, in: J.W. Schmidt (ed.): *Sprachen für Datenbanken*, IFB 72, Springer, 1983

[SS84] H.-J. Schek, M.H. Scholl: *The Relational Model with Relation-Valued Attributes*, Techn. Rep. DVSI-1984-T1, TU Darmstadt, published in: *Information Systems*, Vol. 11:2, 1986

[SW86] H.-J. Schek, G. Weikum: *DASDBS: Concepts and Architecture of a Database System for Advanced Applications*, Techn. Rep. DVSI-1986-T1, TU Darmstadt, 1986

[V86] Verso, J. (pen name for the Verso team): *Verso: A Database Machine Based on Non-First-Normal-Form Relations*, INRIA Report, 1986

[Ul82] J.D. Ullman: *Principles of Database Systems (2nd ed.)*, Computer Science Press, Rockville, MD, 1982