CAN WE USE THE UNIVERSAL INSTANCE ASSUMPTION WITHOUT USING NULLS?

Yehoshua Sagiv
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

ABSTRACT

We claim that the representative instance of [Hol,Va3] is a correct representation of the data stored in a database even when the relations of the database are not the projections of a single universal instance. If no constraint (other than functional and join dependencies) is imposed on the data, then projections of the representative instance cannot always be computed by lossless joins. We show that if the database satisfies a modified foreign-key constraint, then projections of the representative instance can be computed by performing the union of several lossless joins. A class of relation schemes for which no constraint is necessary is characterized, and we show how to compute projections of the representative instance for databases that belong to this class.

1.0 Introduction

The universal instance assumption is an essential assumption in many papers in design theory for relational databases. As pointed out in [FMU], there are two different concepts in this assumption. The most basic concept is the universal relation scheme assumption (also known as the "uniqueness assumption" [Ber]). It asserts that each attribute has a unique role, that is, for any subset of attributes X, there is (at most) one relationship among the attributes of X. This assumption is made explicitly or implicitly in many papers in design theory for relational databases. In particular, it is made in papers dealing with the axiomatization of dependencies, and synthesis and decomposition of relation schemes.

The second and more controversial concept is the universal instance assumption, that is, the assumption that the relations of a database are the projections of a single relation over the set of all the attributes. This assumption is needed in order to define lossless joins [ABU]. There are two versions of this assumption. According to the first, this assumption has to be made only in order to determine whether a join is lossless (i.e., it is only a tool that is used when a database is designed and when queries are evaluated) [FMU]. The second version (the pure universal instance assumption) states that the relations of a database must always be the projections of a universal instance, and null values have to be used in order to satisfy this requirement [HLY,Ko,Li,Ma,Sc].

When a universal instance is assumed, users usually formulate queries having in mind the universal instance rather than the actual relations of the database [KU]. If a given query refers to a set of attributes X, then the first step in evaluating this query is to compute the projection of the universal instance onto X. If the pure universal instance assumption is made, it is suffi-

This research was supported in part by NSF grant MCS-80-03308.

cient to take any lossless join that contains all the attributes of X. If the relations of the database are not the projections of a universal instance, different lossless joins might give different results. In [FMU] this problem is solved by requiring that the join dependency consisting of all the relation schemes be acyclic.

In this paper we assume a universal relation scheme, but not a pure universal instance. In Section 3 we define the representative instance of a database. In [Hol,Val] the representative instance is used to determine whether the database satisfies a set of functional dependencies. We believe that the representative instance correctly describes the information stored in the database even when the relations are not the projections of a universal instance. In the remaining sections we deal with the problem of how to compute projections of the representative instance. In Section 3.1 we show that if no additional constraint is imposed on the data, then projections of the representative instance cannot always be computed by lossless joins. In Section 3.2 we show that if the database satisfies a modified foreign-key constraint, then projections can be computed by performing the union of several lossless joins. In Section 3.3 we characterize a class of relation schemes for which no constraint is necessary. In Section 3 we assume that the only dependencies are functional. In Sections 4 and 5 we extend our results to databases that are required to satisfy a set of functional dependencies and a single join dependency. Essentially, we adopt the view of [FMU] that the only dependency in addition to functional dependencies is a join dependency. However, we do not assume that the join dependency is acyclic.

2.0 Basic definitions

The data in a relational database are stored in tables called relations. The columns of a relation are labeled by distinct attributes. Each attribute has an associated domain of values. The rows or tuples of a relation are mappings from the relation's attributes to their domains. The value of a tuple μ for an attribute A is denoted by $\mu(A)$. A relation scheme is the set of attributes labeling the columns of a relation, and it is usually written as a string of attributes. We often use the relation scheme itself as the name of the relation. A relation is just the "current value" of a relation scheme. A database scheme is a set of relation schemes $R_1,\ldots,R_n$, and a database is a set of relations $r_1,\ldots,r_n$ for the relation schemes $R_1,\ldots,R_n$.

2.1 Relations with null values

In many cases there is a need to represent partial information in the database. If we have a relation over the attributes Manager and Department, and Jones is a manager without a department, then the tuple (Jones,δ) is inserted into this relation. The value δ is a special value, called a null value, and it denotes unknown information. Suppose that there are two managers without a department, e.g., Jones and Smith. There is no reason to assume that they manage the same (unknown) department. In order to distinguish the null value in the tuple (Jones,δ) from the null value in the tuple (Smith,δ), we will mark each null value with a unique subscript and store the tuples (Jones,δ_1) and (Smith,δ_2). Null values with distinguishing subscripts are called marked null [Ko,Ma], and will be used exclusively in this paper. Two null values are equal only if they have the same subscript. We say that tuples μ_1 and μ_2 agree in column A if either both $\mu_1(A)$ and $\mu_2(A)$ are not null and equal or both are null and equal.

Informally, we say that a tuple μ is subsumed by a tuple ν if ν contains all the information stored in μ. When a relation has nulls, one tuple may subsume another tuple even if the two tuples are not identical. We will use the following definition of subsumption that captures some (but not all) the cases in which a tuple can be removed from a relation with null values.(1) A tuple ν sub-

(1) It can be shown that the tuples of a relation r are subsumed by a subset s of r if and only if there is a containment mapping (cf. [ASU1]) from the tuples of r to the tuples of s. (When containment mappings are used for testing subsumption, non-null values are treated as constants and null values are treated as nondistinguished variables.)

sumes tuple μ in a relation r if in every column in which ν and μ disagree, μ has a null value that appears nowhere else in r. We assume that if ν subsumes μ in r, then μ is removed from r.

Example 1: Consider the following relation over the attributes P(player), M(manager), and T(team).

P	T	M
p	δ_1	δ_2
p	δ_3	δ_4
δ_5	δ_3	m

The first tuple agrees with the second tuple in the column for P, and its null values in the other columns appears nowhere else in the relation. Thus, the first tuple is subsumed by the second tuple. However, the second tuple is not subsumed by the first tuple. []

2.2 Relational expressions

In this paper we consider relational expressions over the operators project, (natural) join, and union that are denoted by π, *, and U, respectively. The operands are the relation schemes $R_1,\ldots,R_n$. An expression E is evaluated for a given database by substituting the relations $r_1,\ldots,r_n$ for the relation schemes $R_1,\ldots,R_n$, and applying the operators according to the usual definitions. (When the join is applied, two tuples are joined on a given column only if they agree in this column.)

2.3 Functional dependencies and the chase process

The data stored in a database are usually required to satisfy certain constraints. The most common types of constraints are functional dependencies [Arm,Cl] and join dependencies [MMS,Ris]. The latter type includes multivalued dependencies [BFH,Fag,Li,Zal] as a special case. Suppose that a relation r is required to satisfy a functional dependency (abbr. FD) $X \to A$, where X is a set of attributes and A is a single attribute. This FD can be applied to equate symbols[2] of r in the following way. Suppose that the relation r has tuples μ_1 and μ_2 that agree in all the columns for X but disagree in the column for A. If μ_1 has δ_i in column A and μ_2 has δ_j in column A, then we can replace all occurrences of δ_j in r with δ_i. If μ_1 has a non-null value c in column A and μ_2 has a null value δ_j in that column, then we can replace all occurrences of δ_j with the non-null value c.

Suppose that the relation r is required to satisfy a set F of FD's. We can apply the FD's of F to r until no more symbols of r can be equated. The relation obtained in this way is called the chase of r with respect to F, written $\text{chase}_F(r)$, and it satisfies an FD $X \to A$ of F if and only if there is no pair of tuples that agree in the columns for X and disagree in the column for A. We say that the relation r satisfies F if and only if $\text{chase}_F(r)$ satisfies F. If r satisfies F, then r also satisfies additional FD's that can be inferred by Armstrong's axioms [Arm]. Similarly, given a set of attributes X, we can compute X^+ (i.e., the closure of X with respect to F) using the algorithm of [BB].

2.4 Join dependencies and tableaux

Let r be a relation over a relation scheme R, and let $R_1,\ldots,R_n$ be subsets of R. Tuples $\mu_1,\ldots,\mu_n$ of r are joinable on $R_1,\ldots,R_n$ with a result ν if ν is a tuple on R such that

(1) for all $1 \le i \le n$, ν and μ_i agree on R_i, and

(2) the columns of ν for the attributes of $R - U_{i=1}^n R_i$ have distinct null values that do not appear in r.

The dependency $*[R_1,\ldots,R_n]$ holds in r if whenever tuples $\mu_1,\ldots,\mu_n$ of r are joinable on $R_1,\ldots,R_n$ with a result ν, then ν is subsumed by some tuple of r. If $U_{i=1}^n R_i = R$ (i.e., every attribute of the relation r is in some R_i), then $*[R_1,\ldots,R_n]$ is called a join dependency (abbr. JD). If $U_{i=1}^n R_i \neq R$, then $*[R_1,\ldots,R_n]$ is called an embedded join dependency (abbr. EJD).

(2) Occasionally, we refer to null and non-null values as symbols.

By definition, if r is a relation over $R = U_{i=1}^{n} R_i$, then the relation $*_{i=1}^{n} \pi_{R_i}(r)$ satisfies the JD $*[R_1,\ldots,R_n]$. The expression $*_{i=1}^{n} \pi_{R_i}(R)$ can also be represented by a tableau. A tableau is a matrix whose columns correspond to the attributes of R. The rows of a tableau contain distinguished and nondistinguished variables. Each column has exactly one distinguished variable (that may appear in several rows). The distinguished variable in the i^{th} column is denoted by a_i. Nondistinguished variables are denoted by b_j's.

The tableau T for $*_{i=1}^{n} \pi_{R_i}(R)$ has a row w_i for each R_i. Row w_i contains distinguished variables in the columns for R_i, and distinct nondistinguished variables in the rest of the columns. Let $A_1,\ldots,A_p$ be the attributes labeling the columns of T. The tableau T maps the relation r to the relation T(r) defined by

$\{\mu \mid \mu(A_i) = \rho(a_i)$, where ρ is an assignment of values to the variables of T such that each row of T becomes a tuple of r$\}$

Clearly, $T(r) = *_{i=1}^{n} \pi_{R_i}(r)$. The tableau T for $*_{i=1}^{n} \pi_{R_i}(R)$ can be transformed to a tableau for $\pi_X(*_{i=1}^{n} \pi_{R_i}(R))$ as follows. For every column A_i not in X, replace all occurrences of a_i with a new nondistinguished variable. Tableaux can be minimized, that is, a tableau T can be replaced with an equivalent tableau that has a minimum number of rows [ASU2].

Tableaux are very similar to the tables that represent relations. Consequently, we can apply the chase process to tableaux (distinguished variables are treated as non-null values and nondistinguished variables are treated as null values). The chase of a tableau T with respect to F, written $chase_F(T)$, is also a tableau. Furthermore, if r is a relation such that $r = chase_F(r)$ and r satisfies F, then $T(r) = chase_F(T)(r)$ [ASU1].

2.5 Keys and lossless joins

Let $R_1,\ldots,R_n$ be a database scheme. Each R_i has one or more (explicit) keys. A key K for R_i is a subset of R_i. The relation r_i for R_i cannot have two distinct tuples μ_1 and μ_2 such that $\mu_1(K) = \mu_2(K)$. Clearly, the keys of a relation scheme imply certain FD's. Formally, the relation scheme R_i embodies the set of FD's

$F_i = \{K \rightarrow A \mid K$ is a key of R_i and $A \in R_i\}$

The database scheme embodies the set of FD's $F = U_{i=1}^{n} F_i$.

A relation r over the set of all the attributes R can be stored in the database by assigning $\pi_{R_i}(r)$ to r_i $(1 \leq i \leq n)$. The relation r can be recovered from the relations stored in the database only if the database scheme has the lossless join property [ABU]. An important special case of lossless joins is an extension join [Ho2]. The expression $*_{j=1}^{m} R_{i_j}$ is an extension join of R_{i_1} if $(U_{k=1}^{j} R_{i_k})^+$ contains a key of $R_{i_{j+1}}$ for every $1 \leq j < m$. The extension join $*_{j=1}^{m} R_{i_j}$ is a complete extension join of R_{i_1} if $R_{i_1}^+ = U_{j=1}^{m} R_{i_j}$. If a cover of the FD's is embodied in the relation schemes, then every R_i has a complete extension join consisting of all the relation schemes that are contained in R_i^+.

Example 2: Let ABC, BD, and CDE be the relation schemes (the key of each relation scheme is underlined). ABC * BD is an extension join of ABC. ABC * BD * CDE is a complete extension join of ABC, because $(ABC)^+ = ABCDE$, ABC contains a key of BD, and ABCD contains a key of CDE. But ABC * CDE is not an extension join. []

3.0 The representative instance

The ultimate goal of designing a database scheme has always been a collection of relation schemes $R_1,\ldots,R_n$ that are independent, that is, relation schemes that allow the user to update each relation in the database without having to change the contents of the other relations. Of course, there might be some semantically meaningful constraints (e.g., as in [C2]) that do not allow every possible update. But these constraints should be as limited as possible. We feel that enforcing the universal instance assumption is too restrictive. Clearly, this assumption can always be enforced by

using marked nulls [KU,Ma]. However, this can be done only at the expense of applying the chase process to the universal instance whenever updates are performed on the database. Furthermore, there is a need to store many null values that do not provide any information. These null values are needed only to satisfy the universal instance assumption.

Efficiency is not the only issue. Most relational database systems are not designed to use the chase process. Therefore, there is a need to develop a theory for determining correct access paths when the universal instance assumption is not satisfied. The simplified universal instance assumption of [FMU] is one possible solution. In [FMU] a certain condition is imposed on the database scheme, and it is claimed that under that condition a minimal (in the number of relation schemes) lossless join is a correct access path (even if the relations of the database are not the projections of a universal instance). We will present a different set of conditions and show that queries should be evaluated by performing the union of several lossless joins rather than just one lossless join.

In this section we will define the <u>representative instance</u> of a database $r_1,\ldots,r_n$. The representative instance is defined for every $r_1,\ldots,r_n$ (even if the r_i's are not the projections of a single relation). If $r_1,\ldots,r_n$ are the projections of a universal instance r and the database scheme has the lossless join property, then r is also the representative instance. In [Hol,Val] the representative instance is used to determine whether the database satisfies a set of FD's. We believe that the representative instance is also a correct representation of all the information stored in the database, and queries posed about the contents of the database should be answered with respect to the representative instance.

Let R be the set of all the attributes. A relation r_i can be viewed as a relation over R by adding columns for the attributes in $R - R_i$ that contain distinct null values. Formally, the <u>augmentation</u> of a relation r_i to a relation over R, written $\alpha_R(r_i)$, is

{μ | μ agrees with some tuple of r_i on R_i,
and has distinct null values
(that do not appear in any other tuple)
for the attributes in $R - R_i$}

<u>Example 3</u>: Let r be the relation

A	C
c_1	c_2
c_3	c_4

$\alpha_{ABCD}(r)$ is the relation

A	B	C	D
c_1	δ_1	c_2	δ_2
c_3	δ_3	c_4	δ_4

where $\delta_i \notin \{c_1,c_2,c_3,c_4\}$ for every i. []

Consider a database $r_1,\ldots,r_n$ over a database scheme $R_1,\ldots,R_n$, and let $r' = U_{i=1}^{n}\alpha_R(r_i)$. If there are no dependencies, then r' is the <u>representative instance</u> of the database $r_1,\ldots,r_n$. When dependencies are present, the chase process should be applied to r'. Thus, if the only dependencies are those in the set of FD's F, then the representative instance is $chase_F(r')$. The database $r_1,\ldots,r_n$ satisfies the set of FD's F if the representative instance satisfies F [Hol,Val]. We take the view of [FMU] that all the relevant dependencies in addition to FD's are expressed by a single JD. In Section 4 we define the representative instance assuming that a single JD is the only dependency, and in Section 5 we deal with a set of dependencies consisting of FD's and a JD.

<u>Example 4</u>: Consider the database scheme <u>ABCD</u>, <u>CGDEF</u>, <u>DEFB</u>, and <u>BCF</u> (the key of each relation scheme is underlined). Note that this database scheme is in Boyce-Codd normal form. Suppose that the relation for ABCD is {1112}, the relation for CGDEF is {11111}, the relation for DEFB is {1111}, and the relation for BCF is empty. To obtain the representative instance we have to compute the chase of

A	B	C	D	E	F	G
1	1	1	2	δ_1	δ_2	δ_3
δ_4	δ_5	1	1	1	1	1
δ_6	1	δ_7	1	1	1	δ_8

The null value δ_5 can be replaced with 1 by applying the FD DEF → B to the second and third tuples, and then δ_2 is replaced with 1 by applying BC → F to the first and second tuples. No FD can be applied after that, and so the representative instance is

A	B	C	D	E	F	G
1	1	1	2	δ_1	1	δ_3
δ_4	1	1	1	1	1	1
δ_6	1	δ_7	1	1	1	δ_8

This representative instance (and hence also the database) satisfies the FD's that are embodied in the relation schemes. []

In the above example, the projection of the representative instance onto BCF contains the tuple 111. It may be argued that the tuple 111 over the attributes BCF does not represent a correct information, since the relation for the relation scheme BCF does not contain this tuple. However, if this argument is accepted, then it follows that two distinct relationships between the attributes B, C, and F are stored in the database. One relationship is stored in the relation for the relation scheme BCF, and the other relationship is obtained by the extension join CGDEF * DEFB. The assumption that there is only one relationship between any collection of attributes in the database is essential to many works in design theory for relational databases. Without this assumption the axioms for functional and multivalued dependencies, and the various synthesis and decomposition algorithms cannot be used. Therefore, we believe that the representative instance correctly represents the information stored in the database.

3.1 Computing total projections of the representative instance

In the remainder of this paper we consider a database scheme $R_1,..,R_n$ that embodies a cover F of all the FD's that the data must satisfy, and a corresponding database $r_1,..,r_n$ with a representative instance r. For simplicity's sake, we assume that the relations $r_1,..,r_n$ do not contain null values. (Thus, null values exist only in the representative instance.) We can always decompose a relation scheme into smaller relation schemes, each having the key of the original relation scheme and some of its attributes. Therefore, the only actual restriction implied by our assumption is that null values cannot be stored for the attributes of a key.

The user formulates queries having in mind the representative instance r rather then the individual relations $r_1,...,r_n$. Suppose that the user is formulating a query that refers to a set of attributes X. In order to evaluate this query, we have to compute the projection of r onto X. We assume that if the user is referring to the attributes in X, then he/she is interested only in tuples of r that have non-null values for all the attributes in X. Therefore, the problem addressed in this paper is how to compute the X-total projection of r, i.e.,

{μ | μ is a tuple in $\pi_X(r)$ without any null value}

For example, given the database of Example 4, the ACF-total projection of the representative instance is {111}.

Example 4 illustrates a somewhat surprising fact. An X-total projection of the representative instance cannot always be computed by joining several relations of the database and then projecting onto X. In that example, the ABCDF-total projection of the representative instance is {11121}. However, no expression of the form $\pi_X(*_{j=1}^{m} R_{i_j})$ has as its value the relation {11121} over ABCDF. (Here, $R_{i_1},...,R_{i_m}$ are some of the relation schemes.) In the following sections we will describe two cases in which the problem indicated by Example 4 does not occur.

3.2 The modified foreign-key constraint

The problem presented in the previous section can be solved by imposing a rather simple and

natural constraint on the database. Many papers (e.g., [SmS,Va2,Za2]) assume that databases satisfy the foreign-key constraint. This basic constraint is defined as follows. If a relation scheme R_i contains a set of attributes K that is a key of another relation scheme R_j, and if the relation r_i contains a tuple whose projection onto K is k, then the relation r_j must also contain a tuple whose projection onto K is k.

Example 5: Consider the database scheme ABCDE, CDEF, AFE, and suppose that the relation for each relation scheme is empty. If we insert the tuple 11111 into ABCDE, then the foreign-key constraint requires that we insert into CDEF a tuple whose CDE-value is 111. So, suppose we insert the tuple 1111 into CDEF. Now the database satisfies the foreign-key constraint. []

In the above example, if the relations for ABCDE and CDEF are joined, we get a tuple with an AF-value 11. But the relation for AFE does not contain any tuple with an AF-value 11, although AF is the key of this relation. If the database is intended to satisfy the foreign-key constraint, it is natural to require also that the relation for AFE will have a tuple with an AF-value 11. Moreover, a tuple of ABCDE * CDEF has values for all the attributes of AFE and not just for the key, and so we should require that the projection of that tuple onto AFE will be in the relation for AFE. A constraint that imposes this requirement on a database is defined as follows. Let $R_1,\ldots,R_n$ be a database scheme that embodies a cover F of all the FD's. A database $r_1,\ldots,r_n$ satisfies the modified foreign-key constraint if for every r_i the following is true. If a tuple μ is in r_i, and $R_{j_1},\ldots,R_{j_m}$ are all the relation schemes contained in R_i^+, then there is a tuple ν defined on R_i^+ such that $\nu(R_{j_k})$ is in r_{j_k} $(1\leq k\leq m)$ and $\nu(R_i) = \mu$. An efficient method for enforcing this constraint is described in [KS].

Theorem 1: If the modified foreign-key constraint is satisfied, and each relation r_i satisfies the FD's embodied in its relation scheme, then
(1) the non-null portion of every tuple in the representative instance is included in some extension join, and
(2) the representative instance satisfies all the FD's.

Proof: Let $r' = \bigcup_{j=1}^{n} \alpha_R(r_j)$, and let μ be a tuple in $\alpha_R(r_i)$. Consider the computation of $\text{chase}_F(r')$. By a lemma of [BDB], the only columns of μ that might be changed in the chase process are in R_i^+. By the modified foreign-key constraint and the above lemma, $\text{chase}_F(r')$ can be computed even if we impose the following restriction. Whenever an FD $X \to A$ is applied to a pair of tuples μ_1 and μ_2, one of these tuples must have originated from a relation r_k whose relation scheme embodies $X \to A$. If this restriction is applied, then the tuple μ will have in $\text{chase}_F(r')$ non-null values exactly in the columns for R_i^+. Furthermore, if $R_{i_2},\ldots,R_{i_m}$ are all the relation schemes that embody FD's that are used to replace null values of μ with non-null values, then the non-null portion of μ is contained in the extension join $*_{k=1}^{m} R_{i_k}$, where $R_{i_1} = R_i$.

We now prove that the representative instance satisfies F. Suppose not. That is, there are tuples μ_1 and μ_2 that violate an FD $X \to A$. Let R_k be the relation scheme that embodies $X \to A$. By the modified foreign-key constraint, $\mu_1(R_k)$ and $\mu_2(R_k)$ are in r_k and, therefore, this relation has two distinct tuples with the same value for the key X. []

We now give an algorithm for finding an expression E whose value is the total projection of the representative instance onto a given set of attributes X. The correctness of this algorithm follows from Theorem 1 and its proof.

Algorithm 1
(1) Let S be the set of all the R_i's such that $X \subseteq R_i^+$.
(2) While there are R_i and R_j in S such that $R_i \subseteq R_j^+$, remove R_j from S.
(3) Let $R_{i_1},\ldots,R_{i_m}$ be the relation schemes that remain in S.
(4) For each R_{i_j} let E_{i_j} be an extension join of R_{i_j} that contains X. The expression E is $\bigcup_{j=1}^{m} \pi_X(E_{i_j})$.

Step (2) minimizes the number of extension joins participating in the union. The time complexity of the algorithm is $O(n^3)$.

Example 6: Consider the relation schemes (keys are underlined) DK, HDA, KIA. Let X be KA. After step (1), S = {HDA,KIA}, and that is also the value of S after step (2). An extension join of HDA that contains KA is HDA * DK, and an extension join of KIA that contains KA is KIA. Thus, the expression E is π_{KA}(HDA * DK) U π_{KA}(KIA). []

3.3 Independent relation schemes

We say that the relation schemes $R_1,\ldots,R_n$ are independent if we can perform any insertion or deletion on each relation r_i without having to modify the contents of other relations. Formally, we define the relation schemes $R_1,\ldots,R_n$ to be independent if the following is true. If each relation r_i satisfies the FD's that are embodied in its relation scheme, then

(1) the representative instance satisfies the FD's, and
(2) total projections of the representative instance can be computed by expressions of the form $U_{i=1}^{m}\pi_X(E_i)$, where each E_i is an extension join.

We will give a sufficient condition for a collection of relation schemes $R_1,\ldots,R_n$ that embody a cover of the FD's to be independent. Our condition is much less restrictive than the one implied by [BG], and we believe that many practical applications satisfy it. This condition is also more general than the tree-structured schemas of [Va3].

We say that a relation scheme R_j can add an attribute A to R_i if there is a key K of R_j such that $A \notin K$ and $R_j \subseteq R_i^+$. The relation schemes $R_1,\ldots,R_n$ satisfy the uniqueness condition if and only if for every R_i

(a) no relation scheme (other than R_i) can add to R_i an attribute already in R_i, and
(b) if $A \in R_i^+ - R_i$, then there is a unique R_j that can add A to R_i.

Theorem 2: If the relation schemes $R_1,\ldots,R_n$ embody a cover of the FD's and satisfy the uniqueness condition, then they are independent. Furthermore, if $R_1,\ldots,R_n$ embody a cover of the FD's and each R_i has only one (explicit) key, then they are independent if and only if they satisfy the uniqueness condition.

Example 7: The relation schemes in Example 4 do not satisfy the uniqueness condition, because BCF adds F to CGDEF and F is already in CGDEF. The relation schemes of Example 6 satisfy the uniqueness condition and, hence, are independent. In proof, $(DK)^+$ = DK and DK does not contain any other relation schemes. Similarly, $(KIA)^+$ = KIA and $DK \not\subseteq KIA$, $HDA \not\subseteq KIA$. As for HDA, the relation scheme DK adds K to HDA and $KIA \not\subseteq (HDA)^+$ = HDAK. []

Suppose that $R_1,\ldots,R_n$ embody a cover of the FD's and satisfy the uniqueness condition. We will give an algorithm for constructing an expression E whose value is the total projection of the representative instance onto a given set of attributes X. The expression E is a union of several extension joins that contain X. However, when constructing an extension join that contains X we should be more careful than in Section 3.2.

Example 8: Suppose we have the relation schemes AB, BC, CD, and we need an extension join that contains AC. Only AB has both A and C in its closure, and so we have to consider only extension joins of AB. Both π_{AC}(AB*BC) and π_{AC}(AB*BC*CD) are expressions where a projection onto AC is applied to an extension join of AB. Since no constraint (such as the universal instance assumption or the modified foreign-key constraint) is imposed on the data, the value of the latter expression is contained in but not necessarily equal to the value of the former. []

Since $R_1,\ldots,R_n$ satisfy the uniqueness condition, each R_i (such that $X \subseteq R_i^+$) has a unique minimal (in the number of relation schemes) extension join containing X, and this minimal extension join has to be chosen. Formally, we say that an extension join $E = *_{j=1}^{m}R_{k_j}$ of R_i (i.e., $i = k_1$) is minimal with respect to a set of attributes X if

(1) $X \subseteq U_{j=1}^{m}R_{k_j}$, and

(2) no proper subsequence of $R_{k_1},\ldots,R_{k_m}$ is an extension join of R_i that satisfies (1).

Lemma 1: Let X be a set of attributes such that $X \subseteq R_i^+$. If the relation schemes of the database satisfy the uniqueness condition, then the minimal extension join of R_i with respect to X is unique and can be computed in polynomial time.

We now give an algorithm for computing an expression E whose value is the X-total projection of the representative instance.

Algorithm 2

(1) Let S be the set of all the R_i's such that $X \subseteq R_i^+$.
(2) While there are R_i and R_j in S such that the minimal extension join of R_j with respect to X contains R_i, remove R_j from S.
(3) Let $R_{i_1},\ldots,R_{i_m}$ be the relation schemes that remain in S.
(4) For each R_{i_j} let E_{i_j} be a minimal extension join of R_{i_j} with respect to X. The expression E is $\cup_{j=1}^{m}\pi_X(E_{i_j})$.

Step (2) of the algorithm minimizes the number of extension joins participating in the union. The time complexity of the algorithm is $O(n^3)$.

Example 9: Suppose that the attributes are P(project), D(department), M(manager), L(location), and A(assistant), and the relation schemes are LDP, DPM, and LMA.

Intuitively, the database scheme describes an application in which each project belongs to several departments and is carried out in several locations, but a department can have only one project in each location. In each department participating in a project, there is a manager responsible for that project. Each manager has an assistant in each location. These relation schemes satisfy the uniqueness condition.

Suppose we want to compute the total projection of the representative instance onto LM. After Step (1) of the algorithm, S = {LDP,LMA}. The minimal extension join of LDP with respect to LM is LDP * PDM, and the minimal extension join of LMA with respect to LM is LMA. In Step (2) of the algorithm S is not changed, and so an expression for the LM-total projection of the representative instance is $\pi_{LM}(LDP * PDM) \cup \pi_{LM}(LMA)$.

The result of the above expression is all tuples (l,m) such that either manager m has an assistant in location l or manager m manages some project in location l. Considering the fact that there might be partial information (e.g., a manager with a project in a location where he does not have an assistant, or a manager with an assistant in a location where he does not have a project), the correct answer is indeed given by the above expression. []

4.0 Dealing with join dependencies

Suppose that the database scheme is $R_1,\ldots,R_n$, and the only constraint is the JD $*[R_1,\ldots,R_n]$. The representative instance is $r = *_{i=1}^{n}\pi_{R_i}(r')$, where $r' = \cup_{i=1}^{n}\alpha_R(r_i)$, since r satisfies the given JD. Therefore, the projection onto X is given by $\pi_X(*_{i=1}^{n}\pi_{R_i}(r'))$. Let T' be the tableau for this expression. We can minimize T' as described in [ASU2]. The minimal tableau has to be evaluated with respect to r'. However, we will show how to construct from the minimal tableau T an expression E with the operands $R_1,\ldots,R_n$ whose value is the X-total projection of the representative instance.

Let w_i be a row of T. Suppose that w_i has distinguished and repeated(1) nondistinguished variables exactly in the columns for V. Let $R_{i_1},\ldots,R_{i_q}$ be all the relation schemes that contain V. Construct the expression $E_i = \cup_{j=1}^{q}\pi_V(R_{i_j})$ for row w_i. Let $E_1,..,E_p$ be the expressions constructed in this way for all the rows of T. The X-total projection of the representative instance is given by the expression $\pi_X(*_{i=1}^{p}E_i)$.

Example 10: Suppose that the attributes are P(part), J(project), M(machine), S(supplier), W(warehouse), O(order), and E(employee). The rela-

(1) A nondistinguished variable is repeated if it appears in more than one row.

tion schemes are PJM, PSW, PJSO, JE. The tableau for the expression π_{PE}(PJM * PSW * PJSO * JE) is

P	J	S	M	W	O	E
a_1	b_0	b_1	b_2	b_3	b_4	b_5
a_1	b_6	b_7	b_8	b_9	b_{10}	b_{11}
a_1	b_0	b_7	b_{12}	b_{13}	b_{14}	b_{15}
b_{16}	b_0	b_{17}	b_{18}	b_{19}	b_{20}	a_7

The minimal tableau obtained from the above tableau is

P	J	S	M	W	O	E
a_1	b_0	b_7	b_{12}	b_{13}	b_{14}	b_{15}
b_{16}	b_0	b_{17}	b_{18}	b_{19}	b_{20}	a_7

The expression that corresponds to the first row of the minimal tableau is $E_1 = \pi_{PJ}(PJM) \cup \pi_{PJ}(PJSO)$. The expression that corresponds to the second row is E_2 = JE. The PE-total projection of the representative instance is given by the expression $\pi_{PE}\{(\pi_{PJ}(PJM) \cup \pi_{PJ}(PJSO)) * JE\}$. []

A proof of correctness for this algorithm is based on the technique for minimizing tableaux [ASU2] and is beyond the scope of this paper.

5.0 Functional and join dependencies

We believe that the first step in designing a database scheme is to find relation schemes $R_1,\ldots,R_{n-1}$ in third normal form that embody the FD's [Ber]. Further decompositions of $R_1,\ldots,R_{n-1}$ according to JD's that are not implied by the FD's is undesirable, since some keys will be split between several relation schemes. (As a result, some FD's will no longer be embodied by the database scheme.) However, if for all i $R \not\subseteq R_i^+$ (R is the set of all the attributes), then an additional relation scheme R_n has to be added [BDB]. R_n is a key of R, i.e., $R \subseteq R_n^+$ and no subset of R_n has this property. R_n is needed to ensure that the database scheme has the lossless join property. Since R_n does not embody any FD, we can use JD's to decompose it. We adopt the approach of [FMU] that it is rather easy and natural for the user to specify the relation schemes into which R_n has to be decomposed. Thus, the result of the design process is a set of relation schemes $R_1,\ldots,R_{n-1},R_n,\ldots,R_k$, where the first n-1 relation schemes embody a cover F of the FD's and $R_n' = \cup_{i=n}^{k} R_i$ is a key of R.

5.1 The first case: FD's and a JD.

Suppose that the database must satisfy F, the JD $*[R_1,\ldots,R_k]$, and the modified foreign-key constraint. The representative instance is computed as follows. Let $r' = \cup_{i=1}^{k} \alpha_R(r_i)$. First compute $r'' = chase_F(r')$, and then compute $r = *_{i=1}^{k} \pi_{R_i}(r'')$. By a theorem of [BMSU], the representative instance r satisfies F and $*[R_1,\ldots,R_k]$.

By Theorem 1, during the computation of $chase_F(r')$ a null value is always equated with a non-null value and, therefore, all the null values in r" are distinct. Furthermore, if tuple μ of r' has non-null values exactly in the columns for R_j, then the chase process transforms μ to a tuple with non-null values exactly in the columns for R_j^+.

Now consider the tableau T' for $*_{i=1}^{k} \pi_{R_i}(R)$, and let $T = chase_F(T')$. By a theorem of [ASU1], r = T(r"), since r" satisfies F. Since the relation schemes embody F, all the nondistinguished variables in T are distinct. Furthermore, a row of T' that corresponds to R_j is transformed by the chase process to a row of T with distinguished variables exactly in the columns for R_j^+ [BDB,BMSU]. Thus, the rows of T correspond to complete extension joins of the R_j's. We can now apply the results of Section 4.0 to T provided that we use complete extension joins of the R_j's as the relation schemes.

Example 11: This example is from [FMU]. The attributes are B(bank), A(account), L(loan), and C(customer). The relation schemes are AB, LB, CA, and CL. The tableau for *[AB,LB,CA,CL] is

C	A	L	B
b_1	a_2	b_2	a_4
b_3	b_4	a_3	a_4
a_1	a_2	b_5	b_6
a_1	b_7	a_3	b_8

By applying A → B, we can replace b_6 with a_4. By applying L → B, we can replace b_8 with a_4. No more FD's can be applied after that, and so the chase of the above tableau is

C	A	L	B
b_1	a_2	b_2	a_4
b_3	b_4	a_3	a_4
a_1	a_2	b_5	a_4
a_1	b_7	a_3	a_4

The rows of the above tableau correspond to complete extension joins as follows. The first row corresponds to AB, the second to LB, the third to CA * AB, and the fourth to CL * LB. Suppose we want to compute the CB-total projection of the representative instance. By applying this projection to the above tableau, we get the tableau

C	A	L	B
b_1	b_0	b_2	a_4
b_3	b_4	b_9	a_4
a_1	b_0	b_5	a_4
a_1	b_7	b_9	a_4

and after minimization it becomes

C	A	L	B
a_1	b_0	b_5	a_4

The expression that corresponds to the single row of this tableau is π_{CB}(CA * AB) U π_{CB}(CL * LB).

Similarly, it can be shown that the CALB-total projection of the representative instance is given by the expression CA * AB * CL * LB. []

If $R_1,\ldots,R_k$ satisfy the uniqueness condition (as in the above example), then the modified foreign-key constraint is not required. However, computing an expression E for a total projection is more complicated [Sag].

5.2 The second case: FD's and an EJD

Suppose that the relation schemes $R_1,\ldots,R_{n-1},R_n'$ satisfy the uniqueness condition. The relation scheme R_n' is decomposed into $R_n,\ldots,R_k$ using the EJD $*[R_n,\ldots,R_k]$, and so the database scheme is $R_1,\ldots,R_k$. A representative instance is constructed in two stages. First construct a relation r_n' for R_n' using $r_n,\ldots,r_k$ and the EJD $*[R_n,\ldots,R_k]$, and then construct the representative instance using $r_1,\ldots,r_{n-1},r_n'$ as in Section 3.3. The relation r_n' might have nulls, but these nulls cannot be replaced with non-nulls when the chase of the FD's is computed (because $R_1,\ldots,R_{n-1},R_n'$ satisfy the uniqueness condition). The representative instance as defined here does not necessarily satisfy the given EJD. However, it is not clear that it should satisfy the EJD. A simple case in which the representative instance does satisfy the EJD is discussed in [Sag].

Let X be a set of attributes. A subset K of R_n' is a key for X in R_n' if $X \subseteq K^+$, and for every proper subset K' of K, we have $X \not\subseteq K'^+$.

Lemma 2: If $R_1,\ldots,R_{n-1},R_n'$ satisfy the uniqueness condition, then a set X has a unique key K in R_n'. The key K can be computed in polynomial time.

An expression E for the X-total projection of the representative instance can be constructed as follows. Let S be the set of all the R_j's ($1<j<n$) such that $X \subseteq R_j^+$. Add to S a key K for X in R_n'. Now apply the algorithm of Section 3.3 to the relations $r_1,\ldots,r_{n-1}$ and the relation for K. The relation for K is computed using the EJD $*[R_n,\ldots,R_k]$ and the relations $r_n,\ldots,r_k$ according to Section 4.0.

Example 12: Suppose that the attributes are J(project), S(supplier), P(part), R(price), and E(employee). The relation schemes are SPR, JE, JP, and JS. The only FD is SP → R, and the EJD is *[JE,JP,JS]. Suppose we want to compute the SP-

total projection of the representative instance. A key for SP in JSPE is SP. Thus, the expression for the SP-total projection of the representative instance is E U π_{SP}(SPR). The expression E is computed as follows. The tableau for *[JE,JS,JP] is

J	S	P	E
a_1	a_2	b_1	b_2
a_1	b_3	a_3	b_4
a_1	b_5	b_6	a_4

By applying projection onto SP (i.e., onto the key K of SP in R_n'), we get the tableau

J	S	P	E
b_7	a_2	b_1	b_2
b_7	b_3	a_3	b_4
b_7	b_5	b_6	b_8

Minimizing the above tableau yields

J	S	P	E
b_7	a_2	b_1	b_2
b_7	b_3	a_3	b_4

By the algorithm of Section 4.0, the expression for this tableau is π_{SP}(JS * JP) and so the complete expression is π_{SP}(JS * JP) U π_{SP}(SPR) []

6.0 Conclusions

We have shown that if null values are not used, then some constraints must be imposed on the data to guarantee that queries can be evaluated by lossless joins rather than by the chase process. We have proposed the modified foreign-key constraint as a possible constraint. We have also shown that if the relation schemes satisfy the uniqueness condition, then no constraints are needed. In this case the relations of the database can be updated independently. This result is more general than earlier conditions [BG,Va3]. We have also considered the case where the dependencies are FD's and a JD. We have shown how to apply the optimization techniques of [ASU2] in order to obtain expressions that have as few operands as possible. Finally, we have proposed a method for handling FD's and an EJD. However, this topic requires further investigation.

References

[ABU] Aho, A. V., C. Beeri, and J. D. Ullman, "The Theory of Joins in Relational Databases," ACM Trans. on Database Systems, Vol. 4, No. 3 (Sept. 1979), pp. 297-314.

[ASU1] Aho, A. V., Y. Sagiv, and J. D. Ullman, "Equivalences Among Relational Expressions," SIAM J. Computing, Vol. 8, No. 2 (May 1979), pp. 218-246.

[ASU2] Aho, A. V., Y. Sagiv, and J. D. Ullman, "Efficient Optimization of a Class of Relational Expressions," ACM Trans. on Database Systems, Vol. 4, No. 4 (Dec. 1979), pp. 435-454.

[Arm] Armstrong, W. W., "Dependency Structures of Database Relationships," Proc. IFIP 74, North Holland, 1974, pp. 580-583.

[BB] Beeri, C. and P. A. Bernstein, "Computational Problems Related to the Design of Normal Form Relational Schemas," ACM Trans. on Database Systems, Vol. 4, No. 1 (March 1979), pp. 30-59.

[Ber] Bernstein, P. A., "Synthesizing Third Normal Form Relations from Functional Dependencies," ACM Trans. on Database Systems, Vol. 1, No. 4 (Dec. 1976), pp. 277-298.

[BFH] Beeri, C., R. Fagin, and J. H. Howard, "A Complete Axiomatization for Functional and Multivalued Dependencies in Database Relations," Proc. ACM-SIGMOD Int. Conf. on Management of Data, Toronto, Aug. 1977, pp. 47-61.

[BMSU] Beeri, C., A. O. Mendelzon, Y. Sagiv, and J. D. Ullman, "Equivalence of Relational Database Schemes," Proc. 11th Annual ACM Symp. on Theory of Computing, pp. 319-329, 1979.

[BG] Bernstein, P. A., and N. Goodman, "What Does Boyce-Codd Normal Form Do?," Proc. Int. Conf. on Very Large Data Bases, 1980, pp. 245-259.

[BDB] Biskup, J., U. Dayal, and P. A. Bernstein, "Synthesizing Independent Database Schemas," Proc. ACM-SIGMOD Int. Conf. on Management of Data, 1979, pp. 143-151.

[C1] Codd, E. F., "A Relational Model for Large Shared Data Banks," Comm. ACM, Vol. 13, No. 6 (June 1970), pp. 377-387.

[C2] Codd, E. F., "Extending the Database Relational Model to Capture More Meaning," ACM Trans. on Database Systems, Vol. 4, No. 4 (Dec. 1979), pp. 397-434.

[Fag] Fagin, R., "Multivalued Dependencies and a New Normal Form for Relational Databases," ACM Trans. on Database Systems, Vol. 2, No. 3 (Sept. 1977), pp. 262-278.

[FMU] Fagin, R., A. O. Mendelzon, and J. D. Ullman, "A Simplified Universal Relation Assumption and Its Properties," IBM research report, RJ2900, Nov., 1980.

[Ho1] Honeyman, P., "Testing Satisfaction of Functional Dependencies," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[Ho2] Honeyman, P., "Extension Joins," Proc. Int. Conf. on Very Large Data Bases, 1980, pp.

239-244.

[HLY] Honeyman, P., R. E. Ladner, and M. Yannakakis, "Testing the Universal Instance Assumption," Information Processing Letters, Vol. 10, No. 1 (Feb. 1980), pp. 14-19.

[Ko] Korth, H. F., "A Proposal for the SYSTEM/U Query Language," unpublished memorandum, Stanford University, Stanford, CA, 1980.

[KU] Korth, H. F. and J. D. Ullman, "System/U: a Database System Based on the Universal Relation Assumption," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[KS] Kuck, S. M., and Y. Sagiv, "A Universal Relation Interface for Network Schemas," in preparation.

[Li] Lien, Y. E., "Multivalued Dependencies With Null Values in Relational Databases," Proc. Int. Conf. on Very Large Data Bases, 1979.

[Ma] Maier, D., "Discarding the Universal Instance Assumption: Preliminary Results," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[MMS] Maier D., A. O. Mendelzon, and Y. Sagiv, "Testing Implications of Data Dependencies," ACM Trans. on Database Systems, Vol. 4, No. 4 (Dec. 1979), pp. 455-469.

[Ris] Rissanen, J., "Theory of Relations for Databases - A Tutorial Survey," Proc. 7th Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 64, Springer-Verlag, 1978, pp. 536-551.

[Sag] Sagiv, Y., manuscript in preparation.

[Sc] Sciore, E., "The Universal Instance and Database Design," TR 271, Dept. of Elec. Eng. and Comp. Sci., Princeton University, Princeton, N. J., June, 1980.

[SmS] Smith, J. M. and C. P. Smith, "Database Abstractions: Aggregation," CACM, Vol. 20, No. 6 (Jun. 1979), pp. 405-413.

[Va1] Vassiliou, Y., "Functional Dependencies and Incomplete Information," Proc. Int. Conf. on Very Large Data Bases, 1980, pp. 260-269.

[Va2] Vassiliou, Y. and F. H. Lochovsky, "DBMS Transaction Translation," Proc. COMPSAC 80, Oct., 1980.

[Va3] Vassiliou, Y., "A Formal Treatment of Imperfect Information in Database Management," Technical Report CSRG-123, University of Toronto, Nov., 1980.

[Za1] Zaniolo, C., "Analysis and Design of Relational Schemata for Database Systems," Tech. Rep. UCLA-ENG-7669, Dept. of Comp. Sci., UCLA, July 1976.

[Za2] Zaniolo, C., "Design of Relational Views Over Network Schemas," Proc. ACM-SIGMOD Int. Conf. on Management of Data, 1979, pp.179-190.