PARTITIONING A RELATIONAL DATABASE HORIZONTALLY USING A KNOWLEDGE-BASED APPROACH

D G Shin and K B Irani

Computing Research Laboratory
The University of Michigan
Ann Arbor, Mi 48109

ABSTRACT

This paper concerns estimating the user reference clusters of a database which can be used in partitioning a relational database horizontally during a distributed database design Using the knowledge about the data, the user queries are converted to equivalent queries by a proposed inference procedure The user reference clusters estimated from these revised queries are more precise than those which can be estimated from the original user queries An extension of the language of first-order calculus is developed for the representation of the user queries and the knowledge-base The types of knowledge to be stored are discussed An example illustrates the way inference is carried out, and the soundness of the inference is also discussed

1. Introduction

A suggested practice for distributed database design is to first partition relations horizontally and then distribute the fragments over a given network [RoGo77, TeFr82] The methodology of implementing such an approach has been addressed in only a handful of published works [WoKa83, CeNW83] A major difficulty is that there are no known significant criteria which can be used to partition relations One proposed approach [WoKa83] is to analyze the expressions for the user queries at each site These expressions reveal the user reference clusters (*URC*'s) to the database and these *URC*'s can be used as a means to partition the relations horizontally However, the information contained in the user queries is not sufficient to estimate *URC*'s precisely

In this paper we suggest an approach for better estimating *URC*'s by utilizing not only the user query expressions but also certain knowledge about the data itself The whole approach is formalized by constructing, what we call a Knowledge-Based System (KBS) The KBS consists mainly of two parts (Figure 1), namely, the knowledge base constituting some specific knowledge about the data, and the inference mechanism which is a rule-based inference system [Nils80] The inference mechanism applies the knowledge in the knowledge base to the user queries and generates new equivalent queries The *URC*'s derived from these new queries are more precise than those which can be derived from the original queries Determining horizontal partitions of relations from these estimated *URC*'s is straightforward Therefore, we address the following three questions

(1) How should the user queries and the knowledge be expressed so that the knowledge can be applied to the user queries in a deductive way?

(2) What types of knowledge should be utilized for this purpose?

(3) How should the inference be carried out?

In Section 2, we first formalize a database as a many-sorted structure Then we introduce a first order language called a many-sorted language with aggregate variables (L_Σ) associated with that structure In this section we also discuss how L_Σ is different from the ordinary many-sorted language (L_m) In Section 3 and

Section 4, we show how the user queries to the database and the useful knowledge for KBS are expressed in L_Σ We provide the former in terms of some specific form of formulas of L_Σ and the latter in terms of five axiom schemas In Section 5, we introduce the inference procedure of KBS and illustrate by an example how the inference is made from the axioms and the query expressions We also discuss the soundness of such inference Finally in Section 6, we show how the partitions of each relation are determined from the estimated *URC*'s

2. Many-sorted Language with aggregate variables L_Σ

2.1. Modelling a Database

We first formalize a database as a many-sorted structure Our intention is then to define a first order language associated with the structure so that knowledge about the database and the user demands for the database can be described in this language The way we model the database is similar to what has been described in [GaMi78] where the elementary facts of a real world are viewed as a logical structure and the general facts as a theory whose model is the logical structure

Suppose I and J are a domain index set and a relation index set respectively A database DB is an ordered structure $DB = < \{D_i\}_{i\in I}, \{R_j\}_{j\in J}, \{C_i\}_{i\in I} >$ with associated function $\lambda\ J \rightarrow N^+$ such that for each $i \in I$, D_i is the set of i^{th} sort objects and each D_i constitutes the i^{th} universe of DB , for each $j \in J$, R_j is $\lambda(j)$-ary relation on $\{D_i\}_{i\in I}$ such that $R_j \subseteq D_{i_1} \times \quad \times D_{i_{\lambda(j)}}$, and for each $i \in I$, an element $c \in C_i$ is an element of D_i

Let the relations shown in Figure 2 be a fraction of an auto corporation database Here let the relation *DEALERS* store the information about the dealers with whom the corporation has transactions, let the relation *SALES* be the sales transactions between the divisions of the corporation and the dealers (the divisions of the corporation supply items to the dealers), and let the relation *ITEMS* be the information about the transactions between the corporations and the dealers (the corporation buys resources from some dealers) Let *d#* , *item#* , and *div#* stand for a dealer number, an item number, and a division number, respectively Then the many-sorted structure defined for the auto corporation database, say $DB(Auto)$, is $DB(Auto) = <\{item\#, d\#, \},\{ITEMS, DEALERS, SALES, \},\{B47, V02, \}>$ where *item#* , *d#* , designate sort domains and $B47$, $V02$, are constant symbols which are the members of *item#* sort domain, and so on

Associated with a logical structure as above, we introduce a first-order language L_Σ without function symbols L_Σ is unusual in that the syntactic objects called aggregate variables are additionally featured in its alphabet

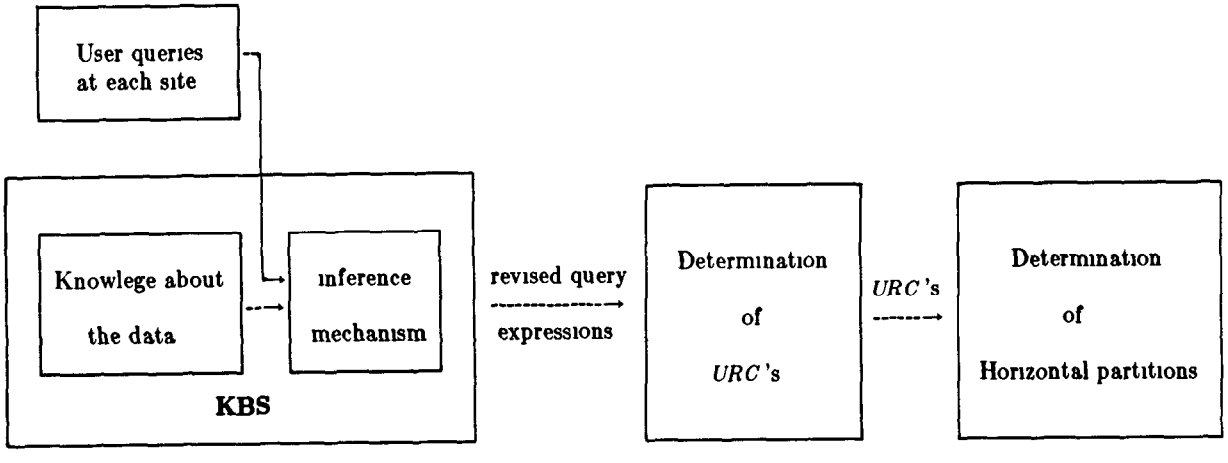

Figure 1 Horizontal Partitioning System

DEALERS

d#	address	d_type
01A	Ann Arbor	51
03A	Dearborn	30
07A	Flint	50
26M	Cleveland	20
33B	Cleveland	30
48B	Rockford	31
55L	Flint	51
65B	Detroit	20
66L	Nile	23
70A	Lansing	70

ITEMS

item#	name	i_type
B47	Eland	bus
C06	white 7	paint
N11	squ 11"	nut
P02	distribu	engin
P03	radiator	engin
S01	In 8080	elect
S02	battery	elect
V01	Astre	sedan
V03	Camaro	sedan
W09	Brat	van
X89	iron 9"	plate

SALES

div#	d#	item#
01AP	01A	V01
01AP	07A	W09
01PP	55L	S01
01PP	07A	P02
02AP	01A	B47
02PP	03A	P02
03PP	01A	P03
03PP	03A	S02
04AP	01A	V03
05AP	55L	V03
05PP	55L	S02

Figure 2 A fraction of an auto corporation database

2.2. Syntax of L_Σ

In L_Σ, we introduce two types of variables, called simple variables and aggregate variables The simple variables of L_Σ are the same as the sort variables of L_m, for instance, the variables $x_i^1, x_i^2,$ which range over the sort domain D_i The aggregate variables are syntactically ordinary sort variables, but semantically they are variables whose ranges of interpretation are restricted to unary relations instead of sort domains Formally stated, if an *aggregate variable of sort* i is of the form $x_i^{\Sigma P}$, then $x_i^{\Sigma P}$ ranges over the unary relation indicated by P which is a subset of the sort domain D_i

Definition 2 1 A *many-sorted language with aggregate variables* L_Σ associated with a database structure DB then consists of the following (1) parentheses (,), and a symbol Σ, (2) constant symbols $c_i^1,$ $, c_i^{n(i)}$, for each $i \in I$, (3) simple variables $x_i^1,$ $, x_i^{m(i)}$ and aggregate variables $x_i^{\Sigma P_1},$ $, x_i^{\Sigma P_{k(i)}}$ for each $i \in I$, where $P_1,$ $, P_{k(i)}$ are predicate symbols, (4) a $\lambda(j)$-ary predicate symbol R_j for each $j \in J$, (5) logical connectives $\neg$, and $\rightarrow$, and (6) universal quantifier $\forall$

Based on this language, the *terms* of L_Σ are the constant symbols, the simple variables, and the aggregate variables The set of *atomic formulas* of L_Σ, $Atom(L_\Sigma)$, is the collection of all the expressions of the form $R_j(t_1,$ $, t_{\lambda(j)})$, where $j \in J$ and $t_1,$ $, t_{\lambda(j)}$ are the terms of L_Σ The set of *well-formed formulas* of L_Σ, $Form(L_\Sigma)$, is defined recursively as (i) if $\alpha \in Atom(L_\Sigma)$, then $\alpha \in Form(L_\Sigma)$, (ii) if $\alpha, \beta \in Form(L_\Sigma)$, then so are $\neg \alpha$, $(\alpha \rightarrow \beta)$, and $\forall x_i^j \alpha$, (iii) nothing else, except those obtained by finite applications of (i) and (ii), is in $Form(L_\Sigma)$ The definable syntactic objects $\cup$, $\cap$, $\leftrightarrows$, and $\exists$, and the standard notions such as *sentences* are also introduced as usual

As an example, let L_Σ associated with the database structure $DB(Auto)$ be $L_\Sigma(DB(Auto)$ Assuming that we will be provided all the symbols needed for $L_\Sigma(DB(Auto))$, $L_\Sigma(DB(Auto)) = < \{ It, De, Sa, \}, \{ B47, V01, \}, \rho >$ where It, De, and Sa are the predicate symbols indicating the relations *ITEMS*, *DEALERS*, and *SALES*, respectively, and $B47$ and $V01$ are the constant symbols which belong to the *item#* sort domain, and ρ is the arity function associated with the predicate symbols

2.3. Interpretation of L_Σ

Given a formula in L_Σ, its interpretation in the structure DB requires the definition of a variable assignment function s as follows

Definition 2 2 For the set V of variables of L_Σ and the universe $\{D_i\}_{i \in I}$ of structure DB, s is an assignment function, s $V \rightarrow \bigcup_i D_i$, such that for a simple variable x_i^j, $s(x_i^j) = a$, where $a \in D_i$, and for an aggregate variable $x_i^{\Sigma P}$, $s(x_i^{\Sigma P}) = a$, where if P^{DB} ($P^{DB} \subseteq D_i$) is the unary relation intended by P in DB, then $a \in P^{DB}$

The validity of each formula is then determined by the following interpretation rules

Definition 2.3 For a formula $\psi \in Form(L_\Sigma)$, the satisfaction of ψ with respect to s in DB is defined as follows

(1) $\models_{DB} R_j(v_0, \ldots, v_{\lambda(j)})[s]$† iff $<s(v_0), \ldots, s(v_{\lambda(j)})> \in R_j$‡

(2) $\models_{DB} \neg\psi[s]$ iff $\not\models_{DB} \psi[s]$

(3) $\models_{DB} \psi_1 \rightarrow \psi_2[s]$ iff if $\models_{DB} \psi_1[s]$ then $\models_{DB} \psi_2[s]$

(4) For simple variable x_i^J, $\models_{DB} \forall x_i^J \psi[s]$ iff for any $a \in D_i$, $\models_{DB} \psi[s(x_i^J \mid a)]$

(5) For aggregate variable $x_i^{\Sigma P}$, $\models_{DB} \forall x_i^{\Sigma P} \psi[s]$ iff for any $a \in P^{DB}$, $\models_{DB} \psi[s(x_i^{\Sigma P} \mid a)]$,

where for variables v_m and v_k,

$$s(v_m \mid a)(v_k) = \begin{cases} s(v_k) & \text{if } v_m \neq v_k \\ a & \text{if } v_m = v_k \end{cases}$$

As a corollary to the definition, the interpretation of $\cup$, $\cap$, and $\exists$ can be also easily defined

2.4. Significance of L_Σ

Previously in [ShIr84], we discussed the correctness of L_Σ, i.e., the theoretical basis for introducing aggregate variables as a part of a first-order language Here we only discuss the significance of L_Σ in our knowledge-based approach

The significance of L_Σ is two fold (1) It permits the introduction of a type of variable, namely, the aggregate variable whose range is restricted to a subset of a sort domain, something which cannot be done in L_m The structure, however, needs to be augmented by the unary predicate which represents the range of the aggregate variable (2) L_Σ provides more compact expressive power than provided by L_m, and consequently, to a certain extent, it becomes possible to develop a simple syntactic matching process with which the knowledge about the data expressed in L_Σ can be applied to the user queries expressed in L_Σ in an inferencing manner

The compactness of L_Σ over L_m can be illustrated by an example Suppose $L_m(DB(Auto))$ is the L_m associated with $DB(Auto)$, and $L_\Sigma(DB(Auto))$ is the L_Σ associated with $DB(Auto)$ Consider the two logically equivalent sentences, (2.1) in $L_m(DB(Auto))$ and (2.2) in $L_\Sigma(DB(Auto))$, namely,

$$\forall x\ \forall y\ \forall z\ (Sa(x,y,z) \cap z = a_1 \cap \cdots \cap z = a_n), \quad (2.1)$$

$$\forall x\ \forall y\ \forall z^{\Sigma P}\ Sa(x,y,z^{\Sigma P}), \quad (2.2)$$

respectively It is clear that (2.2) is expressed in a more compact way than (2.1) To make (2.1) meaningful, however, $DB(Auto)$ is augmented by P^{DB} where $\forall x\ (P(x) \leftrightarrows (x = a_1 \cap \cdots \cap x = a_n))$ How the compact expressive power of L_Σ leads to developing a syntactic matching process useful for our purpose will be described in Section 5

Finally, in order to avoid possible misinterpretation of an aggregate variable in an expression using a bounded quantifier, we show the following example

$$\forall x \in P\quad \psi(x) \quad (2.3)$$

$$\forall x^{\Sigma P}\quad \psi(x^{\Sigma P}) \quad (2.4)$$

Let D_x be the sort domain to which x belongs and $P^{DB} \subseteq D_x$ such that $|P^{DB}| << |D_x|$ Though the interpretations of both are identical, the difference is that the validity of (2.4) is determined after substituting $x^{\Sigma P}$ only for the values of P^{DB}, whereas the validity of (2.3) is determined after substituting x for all the values of D_x

3. Query Representation in L_Σ

The issue of how the user queries should be expressed is whether the user queries can be expressed in an efficient and modular way so that the knowledge about the data can be applied to the user queries in a deductive way With this in mind, we define a class of formulas of L_Σ as follows

Definition 2.4 For $\alpha \in Form(L_\Sigma)$, α is in Σ-*normal* form if, for some $n \geq 0$, α is of the form $\exists v_{i_1} \cdots \exists v_{i_m} \psi(v_1, \ldots, v_n)$, where $\{v_{i_1}, \ldots, v_{i_m}\} \subset \{v_1, \ldots, v_n\}$, such that

(1) v_i, $1 \leq i \leq n$, is a simple or an aggregate variable, and

(2) $\psi(v_1, \ldots, v_n)$ is a conjunction of atomic formulas

† If A is a formula having free variables $v_1, \ldots, v_n$, then we sometimes write $A(v_1, \ldots, v_n)$ for A

‡ To distinguish R_j as an element of DB from that of L_Σ, usually a superscript is used, as in R_j^{DB}, but it is omitted in our context as long as the distinction remains clear

Let $\psi \in Form(L_\Sigma)$ have n free variables Let $DEF(DB,\psi)$ stand for the following n-ary relation defined in DB

$$DEF(DB,\psi) = \{ < a_1, \quad , a_n > \quad \models_{DB} \psi[s] \}$$

where s is a variable assignment function defined in Definition 2 2 We can here prove that with ψ being a Σ-normal form, the minimal capability of a query representation formalism in the sense of [Codd72] is achieved by the expression $DEF(DB,\psi)$ For details see [Shin85]

As an example, suppose a user query to $DB(Auto)$ is "What are the addresses of the dealers which were supplied item# $= B47$, $V01$, or $V03$?" Without using aggregate variables, one way to express the query is $DEF(DB(Auto),\psi_1 \cup \psi_2 \cup \psi_3)$ where

$$\psi_1 = \exists x \ \exists y \ \exists v \ (Sa(x,y,B47) \cap De(y,u,v)),$$

$$\psi_2 = \exists x \ \exists y \ \exists v \ (Sa(x,y,V01) \cap De(y,u,v)), \text{ and}$$

$$\psi_3 = \exists x \ \exists y \ \exists v \ (Sa(x,y,V03) \cap De(y,u,v))$$

Using an aggregate variable, however, the disjunctively conjoined formula $\psi_1 \cup \psi_2 \cup \psi_3$ collapses into a Σ-normal form formula q_1 as follows If J is defined to be $\forall x \ (J(x) \leftrightarrows (x = B47 \cup x = V01 \cup x = V03))$, then

$$q_1 = \exists x \ \exists y \ \exists z^{\Sigma J} \ \exists v \ (Sa(x,y,z^{\Sigma J}) \cap De(y,u,v)) \quad (3\,1)$$

As long as the role of $DEF(DB, \ )$ becomes clear, from here on we consider the formulas in Σ-normal form as the appropriate representations for the user queries

4. Axiomatic knowledge identification

In a knowledge-based system, the types of knowledge to be included in its knowledge base generally depend on the domain of the knowledge to be utilized We postulate that the types of knowledge which are useful for the purpose of KBS fall into the following five types of axiom schemas We first briefly explain what is meant by each axiom schema, and then show the schema representation in L_Σ Examples of each schema are also given later in this section These examples are indicated at each schema description To simplify the expressions, we adopt some abbreviations If A is an index set $A = \{a_1, \quad , a_n\}$, then $\bar{X}_A$ and $Q\bar{X}_A$ are the abbreviations of the sequences $x_{a_1}, \quad , x_{a_n}$ and $Qx_{a_1} \quad Qx_{a_n}$, respectively, where Q is either $\forall$ or $\exists$

(i) Functional Dependency Axiom (FDA) : Functional dependency (FD) in a relation is a well known concept Any type of FD can be expressed in the form of schema discussed below and also any axiom of this schema describes a FD (e g (4 4))

FDA schema Given an n-ary relation R whose attribute index set is $\{1, \quad , n\}$, if there is a FD from $\bar{X}_A$ to $\bar{X}_B$ where A and B are the subsets of $\{1, \quad , n\}$, and $A \cap B$ is not necessarily the empty set, then the FD is expressed as $\forall\bar{X}_A \ \forall\bar{X}_{B'} \ \forall\bar{X}_C \ \forall\bar{Y}_{B'} \ \forall\bar{Y}_C \ (R(\bar{X}_A,\bar{X}_{B'},\bar{X}_C) \cap R(\bar{X}_A,\bar{Y}_B,\bar{Y}_C) \rightarrow (x_{1'} = y_{1'} \cap \quad \cap x_{m'} = y_{m'}))$ where all the variables of $\bar{X}_A$, $\bar{X}_B$, $\bar{X}_C$, $\bar{Y}_{B'}$, and $\bar{Y}_C$ are simple variables, $B' = B - A = \{1', \quad , m'\}$, and $C = \{1, \quad , n\} - (A \cup B)$

(ii) Relationship Axiom (RA) : The knowledge embedded in this schema is that the values of some attributes of a relation are restricted to only the values of the corresponding attributes of some other relation Axioms in this schema describe whether any two relations are joinable, and if they are, then via which attribute(s) they are joinable (e g , (4 5))

RA schema Given an n-ary relation R_1 and an m-ary relation R_2 whose attribute index sets are $\{1, \quad , n\}$ and $\{1, \quad , m\}$ respectively, if the values of $\bar{X}_A$ of R_1 are restricted to the values of $\bar{Y}_B$ of R_2, where $A \subseteq \{1, \quad , n\}$, $B \subseteq \{1, \quad , m\}$, and $|A| \leq m$, then such relationship between these two relations R_1 and R_2 is expressed as $\forall\bar{X}_A \ (\exists\bar{X}_{A'} \ R_1(\bar{X}_A,\bar{X}_{A'}) \rightarrow \exists\bar{Y}_{B'} \ R_2(\bar{X}_A,\bar{Y}_{B'}))$ where all the variables of $\bar{X}_A$, $\bar{X}_{A'}$, and $\bar{Y}_{B'}$ are simple variables, and $A' = \{1, \quad , n\} - A$ and $B' = \{1, \quad , m\} - B$ We say that R_1 is RA-related to R_2

(iii) Inherency Axiom (IA) : This is the type of knowledge which plays the key role in KBS, since URC's can be more precisely estimated by knowing the relationships between the fractions of relations The type of knowledge in this schema consists of the inherited facts specifying how the fractions of relations are interrelated with each other (e g , (4 3))

IA schema Let an n-ary relation R_1 and an m-ary relation R_2, whose attribute index sets are $\{1, \quad , n\}$ and $\{1, \quad , m\}$ respectively, be

related by an axiom of RA on the attributes $\bar{X}_A$, where $A \subseteq \{1, \ldots, n\}$ Then a relationship between some fractions of these two relations R_1 and R_2 is expressed in the form of $\forall \bar{X}_A \, (\exists \bar{X}_{A'} \; \exists \bar{X}_{A^*} \psi(\bar{X}_A, \bar{X}_{A'}, \bar{X}_{A^*}) \rightarrow \exists \bar{Y}_B \; R_2(\bar{X}_A, \bar{Y}_B))$ where possibly some variables of $\bar{X}_{A'}$, $\bar{X}_{A^*}$, and $\bar{Y}_B$ are *aggregate variables*, $A' = \{1, \ldots, n\} - A$, $B = \{1, \ldots, m\} - A$, A^* is some attribute index set of relations RA-related with R_1, and $\psi(\bar{X}_A, \bar{X}_{A'}, \bar{X}_{A^*}) \in Form(L_\Sigma)$ is a conjunction of atomic formulas of L_Σ including $R_1(\bar{X}_A, \bar{X}_{A'})$

Here we notice that any axiom in this schema is a Σ-Horn formula (*Σ-Horn formula* is a variation of Horn formula in which variables in the formula may be aggregate variables)

(iv) Ground Defining Axiom (GDA) : As stated previously whenever a new aggregate variable, say $z_i^{\Sigma P}$, is introduced, the database structure DB is augmented by the set designated by P, i e P^{DB}, so that $\models_{DB} P(z_i)[s]$ iff $s(z_i) \in P^{DB}$ where $P^{DB} \subseteq D_i$ In order to have such an expansion of DB, the members of P^{DB} are explicitly defined in GDA schema in terms of the constants of L_Σ (e g, (4 1))

GDA schema Given a unary predicate P to be introduced, the GDA schema is represented in the form, $\forall z \, (P(z) \leftrightarrows (z = c^1 \cup \cdots \cup z = c^n))$, where c^i, $1 \leq i \leq n$, is a constant symbol of a sort domain to which the aggregate variable accompanying P belongs

(v) Virtual Defining Axiom (VDA) : When defining the set designated by a unary predicate, a formula consisting of already existing relation predicates may be used instead of the constant symbols of L_Σ (e g, (4 6))

VDA schema Given a unary predicate P to be introduced, the VDA schema is represented in the form, $\forall z \, (P(z) \leftrightarrows \alpha(z))$, where $\alpha(z) \in Form(L_\Sigma)$ and z is the only free variable in $\alpha(z)$

In the rest of this section, we show how the knowledge base of KBS is constructed The knowledge base of KBS consists of two levels, which we denote by KB and $KB_{\Sigma H}$ KB is the knowledge base constituting the five types of axioms which have been introduced so far, and $KB_{\Sigma H}$ is a subset of the logical consequences of KB In fact, KB is a proper subset of the complete theory $KB(DB)$ defined on the database structure DB, i e, $KB \subset KB(DB)$ By this we mean that KB is a collection of knowledge selected from $KB(DB)$ which is of interest to the DB designer of a specific application domain In our case, the knowledge needed to estimate the URC's is contained in the five types of axioms

$KB_{\Sigma H}$ is indeed the collection of the axioms which actually take part in the syntactic inference procedure of KBS (which we will introduce in Section 5) $KB_{\Sigma H}$ consists of two types of axioms (i) the IA axioms in KB each of which has a corresponding FDA axiom in KB, and (ii) a class of axioms equivalent to the IA axioms in KB each of which is derived from the IA axioms in KB by using the relevant RA axioms The equivalent axioms are also IA axioms Because both the types of the axioms in $KB_{\Sigma H}$ are IA axioms and the IA axioms are Σ-Horn formulas, $KB_{\Sigma H}$ is called a Σ-Horn knowledge base In the following it will be shown by an example how $KB_{\Sigma H}$ is constructed from KB In this example it will also be clarified why the FDA, RA, GDA, and VDA axioms are included in KB

Let the knowledge provided by a DB designer be "All the dealers who are supplied car items are the car dealers" Let $B47$, $V01$, $V03$, and $W09$ be the only car items, and let 50 and 51 be the only car dealer types If P and Q are defined by

$$\begin{aligned} &\forall z \, (P(z) \leftrightarrows (z = B47 \cup z = V01 \cup z = V03 \cup z = W09)) \\ &\forall z \, (Q(z) \leftrightarrows (z = 50 \cup z = 51)), \end{aligned} \tag{4 1}$$

then a direct recapitulation of the above knowledge in an ordinary many-sorted language is

$$\begin{aligned} \forall y \, (\exists x \; \exists z (Sa(x,y,z) \cap P(z)) \rightarrow \\ \exists u \; \exists v \; (De(y,u,v) \cap Q(v))) \end{aligned} \tag{4 2}$$

By introducing the aggregate variables $z^{\Sigma P}$ and $v^{\Sigma Q}$, (4 2) is equivalently expressed as

$$\begin{aligned} \forall y \, (\exists x \; \exists z^{\Sigma P} \; Sa(x,y,z^{\Sigma P}) \rightarrow \\ \exists u \; \exists v^{\Sigma Q} \; De(y,u,v^{\Sigma Q})) \end{aligned} \tag{4 3}$$

Clearly (4 3) is an IA axiom So (4 3) is an element of KB In the process of generating (4 3), it is required to add the axioms of (4 1) in KB as GDA axioms in order to make P and Q meaningful predicate symbols

At this time it is not certain whether (4 3), included in KB, is a useful IA axiom for our purpose This requires the presence of a corresponding FDA axiom in KB Suppose the following FDA axiom from *d#* to *d_type* in *DEALERS* is included in KB

$$\forall x\ \forall y\ \forall z\ \forall y'\ \forall z'(De(x,y,z) \cap De(x,y',z') \rightarrow z=z') \quad (4\ 4)$$

Then (4 3) in conjunction with (4 4) assures us that using (4 3) in the inference procedure of KBS will not lead to an incorrect identification of *URC*'s We illustrate this by an example later in Section 5 For this reason we include in $KB_{\Sigma H}$ only the IA axioms which have their corresponding FDA axioms in KB Thus (4 3) is an element of $KB_{\Sigma H}$

We now show how $KB_{\Sigma H}$ is expanded by adding new IA axioms to enlarge the class of user queries that can be handled by the KBS These new axioms are equivalent to the original IA axioms included in $KB_{\Sigma H}$, and they are generated in conjunction with some relevant RA axioms in KB Suppose there is a relationship between *SALES* and *ITEMS* via *item#*, which is expressed by a RA axiom such as

$$\forall z\ (\exists x\ \exists y\ Sa(x,y,z) \rightarrow \exists w\ \exists t\ It(z,w,t)) \quad (4\ 5)$$

If the aggregate variable shown in the antecedent of (4 3) is unraveled, (4 3) is equivalently expressed as $\forall y(\exists x\ \exists z(Sa(x,y,z) \cap P(z)) \rightarrow \exists u\ \exists v^{\Sigma Q} De(y,u,v^{\Sigma Q}))$ The unary predicate P which was once defined in terms of constants can now be defined in terms of the predicate It Let R be a unary predicate whose interpretation in the structure DB is $R^{DB} = \{sedan, bus, van\}$ Then from the fact that $P^{DB} = DEF(DB, \exists w\ \exists t^{\Sigma R}\ It(z,w,t^{\Sigma R}))$, the meaning of P is introduced by a VDA axiom

$$\forall z\ (P(z) \leftrightarrows \exists w\ \exists t^{\Sigma R}\ It(z,w,t^{\Sigma R})) \quad (4\ 6)$$

where the meaning of the predicate R is defined by the GDA axiom $\forall z\ (R(z) \leftrightarrows (z=sedan \cup z=bus \cup z=van))$ Now by using (4 6), (4 3) can be equivalently expressed as

$$\forall y\ (\exists x\ \exists z\ \exists w\ \exists t^{\Sigma R}(Sa(x,y,z) \cap It(z,w,t^{\Sigma R})) \rightarrow \exists u\ \exists v^{\Sigma Q}\ De(y,u,v^{\Sigma Q})) \quad (4\ 7)$$

Here, (4 7) is again an IA axiom, and so, (4 7) is included in $KB_{\Sigma H}$ In Section 5, we show by an example why in addition to (4 3), its equivalent axiom (4 7) is also needed to be included in $KB_{\Sigma H}$ We notice that though $KB_{\Sigma H}$ contains only IA axioms (which are all Σ-Horn formulas), the other types of axioms have been indirectly embedded in the construction of $KB_{\Sigma H}$

5. Inference Procedure and Its Illustration

Here we describe how the knowledge about the data, i e, $KB_{\Sigma H}$, is applied to the user queries, i e, query expressions in Σ-normal form, via the inference procedure of KBS We abbreviate the inference procedure by the symbol "$\vdash\!\!\rightarrow$" "$\vdash\!\!\rightarrow$" requires two preliminary steps to be taken for the formulas in $KB_{\Sigma H}$ and the query expressions in Σ-normal form Each formula in $KB_{\Sigma H}$ is to be converted into an existential quantifier free form, and each query expression in Σ-normal form is to be existentially closed and assumed valid in the structure DB Once these steps are performed, all the quantifiers can be omitted from the formulas in $KB_{\Sigma H}$ and the query expressions This is possible because the formulas in $KB_{\Sigma H}$ are only universally quantified and the query expressions are only existentially quantified

The query expressions in Σ-normal form are existentially closed as usual The formulas in $KB_{\Sigma H}$ are converted into existential quantifier free forms by the usual procedure called Skolemization The Skolemization for the formulas of L_Σ differs from the ordinary Skolemization only by the fact that here when a Skolem function is introduced, its range needs to be restricted to a unary relation which is the same as the range of the variable to be replaced by the function

The Skolemization step needs no justification as long as the Skolemized formulas are equivalent to the formulas prior to Skolemization The other step, however, needs justification because when the query expressions are existentially closed, the resulting formula may no longer be valid in the structure if the solution to the query is empty This step is justified, however, by the fact that the existentially closed formula is used only to determine the fractions of the database referred to by the query Whether a query has an empty solution is irrelevant for determining the fractions of the database referred to by the query

Once the preliminary steps are made, "$\vdash\!\!\rightarrow$" manipulates the matrices of the Skolemized formulas in

$KB_{\Sigma H}$ with the matrices of the existentially closed query expressions In order to describe "$\vdash\!\!\rightarrow$", we first define two notions, namely *"match"* and *"restrictable"* Let $KB^m_{\Sigma H}$ be the collection of all the matrices of the Skolemized formulas in $KB_{\Sigma H}$ Let Q^m_c be the collection of all the existential closed query expressions which we are concerned with, for example, the user provided queries from which we are trying to derive the *URC*'s For $q \in Q^m_c$, let $SUB(q)$ stand for the collection of all the subformulas of q Each formula in $SUB(q)$ is again a conjunction of atomic formulas Since any formula in $KB_{\Sigma H}$ is of IA schema, it follows that any formula in $KB^m_{\Sigma H}$ is of the form $\psi \rightarrow R(t_1, \quad, t_n)$ where ψ is a conjunction of atomic formulas and $R(t_1, \quad, t_n)$ is an atomic formula with R being a relation predicate and some of the terms among $t_1, \quad, t_n$ being Skolem functions We use this formality in defining the following notions

We say that for $q \in Q^m_c$ and $K_j \in KB^m_{\Sigma H}$ with K_j of the form $\psi_j \rightarrow R(t_1, \quad, t_n)$, some $q_i \in SUB(q)$ ***matches*** K_j if $DEF(DB, q_i) \subseteq DEF(DB, \psi_j)$ q is ***restrictable*** by K_j if the following two conditions are satisfied

(1) There is $q_i \in SUB(q)$ which matches K_j, and

(2) for the consequent $R(t_1, \quad, t_n)$ of K_j, there is an atomic formula $R_q(t_1^q, \quad, t_n^q)$ in q such that (i) R and R_q are identical relation predicates, and (ii) there is a variable t_i^q and a Skolem function t_i such that $Ran(t_i) \subset Ran(t_i^q)$, where by $Ran(t)$ we mean the range of the term t

We call (R, R_q) *a restriction pair* If q is restrictable by K_j, we restrict q by K_j using the following process For each restriction pair (R, R_q), if the variable t_i^q in R_q and the Skolem function t_i in R satisfy the relationship $Ran(t_i) \subset Ran(t_i^q)$, then substitute t_i^q by a variable v whose range $Ran(v) = Ran(t_i)$ We denote the restricted q by $q \mid K_j$ "$\vdash\!\!\rightarrow$" applies $KB^m_{\Sigma H}$ to a $q \in Q^m_c$ to produce q^* which is equivalent to q q^* shows more precise *URC*'s than what q does The inference procedure "$\vdash\!\!\rightarrow$" is the following

Inference Procedure "$\vdash\!\!\rightarrow$"

Step 1 Let $q^* = q$, $W = SUB(q)$, and go to Step 2

Step 2 Let q_i be an element of W, and go to Step 3

Step 3 Let $MATCH(q_i)$ be all the formulas in $KB^m_{\Sigma H}$ which match with q_i If $MATCH(q_i)$ is empty, go to Step 5, otherwise go to Step 4

Step 4 Do while $MATCH(q_i)$ is not empty,

1 let K_j be an element of $MATCH(q_i)$,

2 $MATCH(q_i) = MATCH(q_i) - K_j$, and

3 let $q^* = q^* \mid K_j$ only if q^* is restrictable by K_j,

and go to Step 5

Step 5 Let $W = W - q_i$ If W is empty, stop, otherwise go to Step 2

The above procedure always stops at Step 5 with q^* being a revised version of $q \in Q^m_c$ When it is convenient, we abbreviate the whole procedure by $KB^m_{\Sigma H} \cup q \vdash\!\!\rightarrow q^*$ In the following we illustrate "$\vdash\!\!\rightarrow$" by an example

Suppose the query we are concerned with is q_1 in (3 1) The existential closure q^c of q_1 is then

$$q^c = \exists x\ \exists y\ \exists z^{\Sigma J}\ \exists u\ \exists v(Sa(x,y,z^{\Sigma J}) \cap De(y,u,v)) \quad (5\,1)$$

We now show how the IA axiom (4 3) in $KB_{\Sigma H}$ is applied to q^c of (5 1) to derive q^* in a purely syntactic way by "$\vdash\!\!\rightarrow$" First, (4 3) is converted to a prenex normal form such as

$$\forall y\ \forall z\ \forall z^{\Sigma P}\ \exists u\ \exists v^{\Sigma Q}\ (Sa(x,y,z^{\Sigma P}) \rightarrow De(y,u,v^{\Sigma Q}))$$

The above formula is Skolemized to

$$\forall y\ \forall z\ \forall z^{\Sigma P}\ (Sa(x,y,z^{\Sigma P}) \rightarrow De(y,g(y,x,z^{\Sigma P}),f^Q(y,x,z^{\Sigma P}))), \quad (5\,2)$$

where $g(y,x,z^{\Sigma P})$ and $f^Q(y,x,z^{\Sigma P})$ are the Skolem functions which replace u and $v^{\Sigma Q}$, respectively We notice that the range of $f^Q(y,x,z^{\Sigma P})$ is denoted by the superscript Q which is the range of $v^{\Sigma Q}$ Here, (5 2) clearly shows how the relations *SALES* and *DEALERS* are related fragment by fragment, namely *CAR_SALES* of *SALES* and *CAR_DEALERS* of *DEALERS* Now let the matrices of (5 1) and (5 2) be q and K_j, respectively The followings hold $Sa(x,y,z^{\Sigma J})$ in q matches with the antecedent $Sa(x,y,z^{\Sigma P})$ of K_j, i e,

$$DEF(DB(Auto), Sa(x,y,z^{\Sigma J})) \subseteq DEF(DB(Auto), Sa(x,y,z^{\Sigma P}))$$

Also, q is restrictable by K, since there is a restriction pair $(De(y,g(y,x,z^{\Sigma P}),f^{Q}(y,x,z^{\Sigma P})), De(y,u,v))$ where $Ran(f^{Q}(y,x,z^{\Sigma P})) \subset Ran(v)$ Therefore, by substituting v in $De(y,u,v)$ by the variable $w^{\Sigma Q}$ whose range is identical to that of $f^{Q}(y,x,z^{\Sigma P}))$, q' is concluded to be

$$q' = Sa(x,y,z^{\Sigma J}) \cap De(y,u,w^{\Sigma Q}) \qquad (5\ 3)$$

The URC's indicated by q' in (5 3) are the defined relations $DEF(DB(Auto),Sa(x,y,z^{\Sigma J}))$, and $DEF(DB(Auto),De(y,u,w^{\Sigma Q}))$, whereas those indicated by q are the defined relations $DEF(DB(Auto),Sa(x,y,z^{\Sigma J}))$, and $DEF(DB(Auto),De(y,u,v))$ It is clear that q' shows more precise URC's than what q does

In general, for any symbolic manipulation procedure designed to carry out inference, it needs to be justified that whether the result obtained syntactically is indeed valid semantically What matters is the soundness issue of "$\vdash\!\!\rightarrow$" In addition, it needs to be justified that q and q' are equivalent These two issues are formalized by the following two theorems

Theorem 5 1 The Soundness of "$\vdash\!\!\rightarrow$"

For $q \in Q_c^m$, let $KB_{\Sigma H}^m \cup q \vdash\!\!\rightarrow q'$ If q_c and q_c' is the existential closures of q and q', respectively, then $KB_{\Sigma H}^m \cup q_c \models_{DB} q_c'$

Theorem 5 2 Equivalence of q and q'

For $q \in Q_c^m$, let $KB_{\Sigma H}^m \cup q \vdash\!\!\rightarrow q'$ Then $DEF(DB,q) = DEF(DB,q')$

Proofs in full details can be found in [Shin85] Here we simply argue their correctness with respect to the previous example

As far as the soundness of "$\vdash\!\!\rightarrow$" is concerned, it can be easily argued as follows as long as we know that all the dealers who are supplied the cars are the car dealers and that there are some dealers who have been supplied items $B47$, $V01$, or $V03$ which are the cars, it is valid to conclude that there are some dealers who are car dealers

The equivalence of q an q' can be proved in conjunction with a FDA axiom in KB which corresponds to the IA axiom used in "$\vdash\!\!\rightarrow$" In our example, since the inclusion of the IA axiom (4 3) in $KB_{\Sigma H}$ has been accompanied by the FDA axiom (4 4) in KB, it can be easily shown that

$$DEF(DB(Auto), \exists u\ \exists v^{\Sigma Q}\ De(y,u,v^{\Sigma Q})) \cap$$

$$DEF(DB(Auto), \exists u\ \exists v^{\Sigma Q'}\ De(y,u,v^{\Sigma Q'})) = \phi$$

where $\forall x(Q'(x) \leftrightarrows \neg Q(x))$ In other words this means that we can safely interpret (5 3) more restrictively as "All the dealers who are supplied items $B47$, $V01$, or $V03$ are car dealers"

Finally we illustrate the reason why $KB_{\Sigma H}$ is needed to be expanded by the axioms equivalent to its IA axioms Suppose the user query q_1 in (3 1) had been equivalently given as q_2,

$$q_2 = \exists z\ \exists w\ \exists t^{\Sigma R}\ \exists x\ \exists y\ \exists v\ (\ It(z,w,t^{\Sigma R})$$

$$\cap\ Sa(x,y,z) \cap De(y,u,v))$$

where $\forall x\ (R(x) \leftrightarrows (x = sedan \cup x = bus \cup x = van))$ Let q be the matrix of the existential closure of q_2,

$$q = It(z,w,t^{\Sigma R}) \cap Sa(x,y,z) \cap De(y,u,v) \qquad (5\ 4)$$

Then no subformulas of q matches (5 2) although $It(z,w,t^{\Sigma R}) \cap Sa(x,y,z)$ semantically matches (5 2) However, we can still derive the revised version q' of q in (5 4) by using the IA axiom (4 7) which has been previously shown equivalent to (5 2) After converting (4 7) to the prenex normal form, similar procedure can be applied to (5 4) and (4 7), as had been done for (5 1) and (5 2), to conclude q',

$$q' = It(z,w,t^{\Sigma R}) \cap Sa(x,y,z) \cap De(y,u,w^{\Sigma Q}) \qquad (5\ 5)$$

We see that (5 5) shows more precise URC's than what (5 4) does, and the argument similar to the previous one can be provided to show that (5 4) and (5 5) are equivalent queries

6. Horizontal Partitioning

In this section we discuss the following two issues first, how do we use the estimated URC's for partitioning the relations?, and second, how should we interpret the partitions of the relations obtained by our approach? The first issue is straightforward We view each revised version of a query, say q', as a way of obtaining a bipartition of each relation referred to by q' That is,

a relation being referred to by q' is divided into two fragments, one part needed to answer q' and the other not needed For instance, in our example, we obtain from the revised query expression (5 3) a bipartition of *DEALERS* , namely, *CAR_DEALERS* and *NON_CAR_DEALERS* , and a bipartition of *SALES* , namely, *CAR_SALES* and *NON_CAR_SALES* (see Figure 3) We notice that the bipartition of *DEALERS* is not derivable from the original query expression (3 1)

In fact, more than one query possibly refer to each specific relation in the database Let $Q'(R)$ be the collection of the restricted versions of queries referring to the relation R One way to obtain a partition of the relation R is to intersect all the possible bipartitions, each of which is obtained from each restricted query expression in $Q'(R)$

Various data allocation algorithms such as [MoLe77, IrKh79, Aper81, CeNW83] can be used to determine an optimal or suboptimal dispersion of the data by treating the fragments as the unit objects of distribution The fragments can be flexibly distributed over a network

7. Conclusion

We have described a knowledge-based approach in which *URC*'s is derived from the user queries to the database and the knowledge about the data In order to describe the user queries and the knowledge, the ordinary many-sorted language is extended In this extended language, the user queries are expressed in a specific form, and the knowledge useful for our purpose is identified by five types of axiom schemas The knowledge is applied to each query expression via an inference mechanism to derive a revised query expression From the revised query expressions, *URC*'s are estimated Horizontal partitioning is based on the estimated *URC*'s

8. Aknowlegement

We thank to the refrees for their helpful comments

This work was supported in part by Air Force Office Scientific Research under contract F49620-82-C-0089

CAR_DEALERS

d#	address	d_type
01A	Ann Arbor	51
07A	Flint	50
55L	Flint	51

DEALERS

NON_CAR_DEALERS

d#	address	d_type
03A	Dearborn	30
26M	Cleveland	20
33B	Cleveland	30
48B	Rockford	31
65B	Detroit	20
66L	Nile	23
70A	Lansing	70

CAR_SALES

div#	d#	item#
01AP	01A	V01
01AP	07A	W09
02AP	01A	B47
04AP	01A	V03
05AP	55L	V03

SALES

NON_CAR_SALES

div#	d#	item#
01PP	55L	S01
01PP	07A	P02
02PP	03A	P02
03PP	01A	P03
03PP	03A	S02
05PP	55L	S02

Figure 3 Bipartitions of *DEALERS* and *SALES*

BIBLIOGRAPHY

[Aper81] Apers, P M G "Redundant allocation of relations in a communication network", Proc of the Fifth Berkeley Workshop on Distributed data management and computer networks, Lawrence Berkeley Lab , 1981

[Codd72] Codd, E F "Further normalization of the data base relational model", Data Base Systems (R Rustin, ed), Prentice Hall, pp 65-98, 1972

[CeNW83] Ceri, S, Navathe, S, and Wiederhold, G, "Distributed design of logical database schemas", IEEE Tran on Software Engineering, Vol SE-9, No 4, pp 487-504, July, 1983

[GaMi78] Gallaire, H and Minker, J (eds), "Logic and Databases", Plenum Press, 1978

[IrKh79] Irani, K B and Khabbaz, N G "A model for combined communication network design and file allocation for distributed databases", Proc of the First Int Conf on Distributed Computing Systems, Huntsville, Al , pp 15-21, Oct 1979

[MoLe77] Morgan, H L and Levin, K D "Optimal program and data locations in computer networks", CACM, Vol 32, No 5, May, 1977

[Nils80] Nilsson, N J "Principles of Artificial Intelligence", Tioga Pub Co , 1980

[RoGo77] Rothnie, J B and Goodman, N "A survey of research and development in distributed database management", Proc of the Int Conf on VLDB, Tokyo, pp 48-62, Oct , 1977

[Shin85] Shin, D G "An Extension of a First-Order Language and Its Applications", Ph D Dissertation, University of Michigan, Ann Arbor, Being edited, 1985

[ShIr84] Shin, D G and Irani, K B "Knowledge representation using an extension of a many-sorted language", Proc of the First Conf on Artificial Intelligence Applications, Denver, Colorado, Dec 1984

[TeFr82] Teory, T J and Fry, J P "Design of Database Structure", Prentice-Hall, 1982

[Wang52] Wang, H "Logic of Many-Sorted Theories", The Journal of Symbolic Logic, Vol 17, No 2, June, 1952

[WoKa83] Wong, E and Katz, R H "Distributing a database for parallelism", Proc of SIGMOD 83, San Jose, Calif , pp 23-29, May, 1983