Meta-Database Architecture for Relational DBMS

Kn. I. Kilov and I. A. Poppva

Programmirovanie, Vol. 1, 1981

UDK 681.3

The architecture of an implementable relational DBMS is discussed, the meta-database for which is organized in the form of a set of text tables. Methodology for working with text is shown (including automatic coding), as well as semantic text control. A schematic example of a meta-database is shown.

At the start of the 1960's -- the time of the development of COBOL -- the need to separate data attribute description from the description of data processing procedures became apparent.

Recently it has become obvious that the nucleus of any database management system (DBMS) must be a fund of system information concerning the database [1]. To this fund, in particular, belong the database schemas (conceptual, internal and external) and the dictionaries of textual information, which are together called the meta-database.

For a relational DBMS, textual information includes not only the names of attributes, domains, and relationships accompanied by their schematic features (and the synonyms of these names), but also the names of such "non-standard" entities as the document forms supported by the data input system, interpretable conditions for database semantic integrity [2], information on users etc. In this regard, a separate problem arises in organizing storage and text processing for constructing the meta-database. A solution to this problem is used in developing a relational hybrid DBMS [3-5].

The idea to separate text information according to storage type has already been proposed, by way of example, for working with dictionaries used for processing messages in natural language in factographic information retrieval systems [6], as well as for relational DBMS [7], in order to provide convenient storage and effective memory utilization. However the method described below for processing text presupposes not only the effective "convolution" of text values, but also (this has apparently not been previously proposed) supports the reliability and semantics of text data in implementing a relational DBMS.

Texts in a database are organized in the form of system name tables, i.e. grouped into (text) domains. Text values, which are names of schema elements (attributes, domains, relations, tests etc.) are stored in special system tables, which are called schema tables. All text tables in the database are organized in the same way, i.e. identical in construction. However schema tables, in addition to their basic component, contain additional information on actual entities from the given table. Using such an approach, the meta-database, as recommended in [1], is an integral component part of the database. Thus, as compared for example to [8], the meta-database is actually used for providing data integrity.

A traditional stage in constructing an information processing system is still "classification and coding". The system being discussed here implements automatic coding of all text values, thereby making it possible to avoid manual coding. For each text, a three-byte hash code ranging from 0 to 16777215 ($=2^{24}-1$) is generated. The fourth byte serves as the collision descriminator and in this way, each text is given a unique four-byte code in its text table (in its domain). (Note that in our hash—function test, there were no collisions in coding domains consisting of 10000 actual texts with a length of from 1 to 256 bytes. The test utilized documentation texts for the DBMS being discussed here, which were stored in direct access volumes, and set up for editing and printing.) In all user relations (and in other database tables) are stored not texts but their four-byte codes. In so far as all numerical attribute values stored in the relations also take up a fixed number of bytes, it is clear that all user relations have identical structure -- lines of constant length.

This kind of database organization, i.e. user relations, is very convenient for executing any operation on the database. This simplicity is attained because of the "complex" organization of meta-database text tables. The records of these tables are of variable length, which is determined by the length of a text value, the number of declared synonyms, and other information.

Self-contained text storage makes it possible to "explain" to the system the semantic equivalence of syntactically different text values, i.e. introduce the concept of synonyms. Examples of these values include "yarko-zelenyi" ["bright-green"], "yarkozelenyi" ["brightgreen"], and "yark/zel" ["br/gr"]. This type of text variation is natural in inputting information from actual (and not specially set-up for punching) documents. Thus all semantically equivalent but syntactically non-identical text values must be represented uniquely in the database -- by some kind of reference [etalonnyi] value. On the other hand, in order to inform the system of the complete set of values equivalent to the given value, certain special facilities are essential -- a synonymic apparatus. Terms that are convenient and natural for the actual user are used as synonyms. A sufficient number of synonyms (up to 254) are allowed for each reference text.

The system described here for processing and storing text values has made it possible to maintain semantic integrity of text domains, including database security in the event of errors in the texts. For this reason, before loading the information (for example, from documents) the user relations must be independently filled with relevant text domains by reference text values and their appropriate synonyms. Dynamic modification and augmentation of these domains is likewise subsequently executed independently. Thus it is assumed that all newly entered texts (the first time being the only time) must be visually checked.

Text domain support in actual practice is a responsibility of the administrator for the actual application; for this purpose, he has at his disposal special utilities and forms for information input.

Any text attribute value entered into the database is checked by the system for its existence (in the form of a reference [etalon] or synonym) in the relevant domain for the given attribute. A text not found in the domain is considered to be an error. Upon finding the text in the domain, the synonymic apparatus always localizes the reference text and its system code. It is the actual code that is the representative of the whole set of semantically equivalent values in relational DBMS operations that interpret the calculation of the relations.

As a rule, it is only possible to get a reference value from the database. However, if necessary, one of the synonyms may be declared to be a "verbal name" which may be given instead of the reference, by special request. For example, the reference "Latvian SSR" may have the synonyms "Latvijas PSR", "Latvia, LSSR", and the verbal name "Latvian Soviet Socialist Republic". If for each reference value a verbal name is given in another natural language, then the entire system may work equally well in two languages.

It should be noted that in the DBMS described here, the concept of domain is used only in connection with the organization of storage and text processing. Domains are treated as named abstract (interpreted) data types (compare with [9], for example) and in the meta-database for each domain, all information describing the domain is stored. It is possible to give the range for numerical base types, for example, complete divisibility for the base type "integer" etc. Thus domains with completely corresponding characteristics but different names are treated as differently interpreted data types (a traditional example are the types "height" and "weight": base type "integer", range 1-250), which makes it possible to reject semantically illegical operations on relations.

With the meta-database organization as described here, it is possible to automatically test data entered into the system for correspondence to all characteristics of the relevant domains. In this way, stored data is dynamically matched with its description in the meta-database.

To illustrate the architecture described here, an example diagram is given (cf. drawing), in which the technical details are omitted, with hash-code text value (only for illustrative purposes) considered to be the hash-code's first three symbols. (Actual hash codes are assigned in the following way. Prior to hashing, the text is normalized, i.e. all semantically valueless gaps are removed from the text. Then the text is decomposed into three-byte fields, which are treated as integer numbers. These numbers are summed up, to which is added the length of the text. The result is divided by the integer

number closest to $2^{24}-1$ which is the product of two prime numbers. The remainder from the civision is the hash code.) In all the text tables for the reference value ("e"), a list of synonyms is shown if they exist, as well as indication of a reference for the synonym ("s"). The inverse list of relations, to the schema of which belorgs the given domain, is shown in the domain table, and the list of attributes and their corresponcing domains that belong to the schema of the given relation is shown in the relation table. (For illustrative purposes, it is assumed that the names of attributes correspond to the names of their domains; this is generally not so.) Other schema fielos in tne system tables are not shown in the example so as not to overload the diagram (such as a list of documents from which the actual relations may be filled; a list of parameters describirg tne domain as an abstract (interpretable) oata type etc.).

The oatabase schema incluoes:

domains:

year record was set take integer values (base type Ts)
number of student card

type of sport
stadium take text values (base type T)
student surname

relations:

SPORTING ACTIVITIES (number of student card, stuoent surname, type of sport, stadium);

RECORD HOLDER (type of sport, year of record set, student surname, number of student card).

Basic table organization and organization of information input from documents is discussed in more detail in [4, 5]. In the proposed relational DBMS architecture, system tables, i.e. domain description tables, relation description tables, text attribute value tables, inverse tables etc., are the actual meta-database. With this kind of organization, it is comparatively easy to modify schema information on the database (which is unavoidable), meanwhile preserving application software indepenoence from the data as recommenced in [1]. (In particular, automatic "tuning" of document input programs [4] is provided for changes in schema characteristics related to the relevant domains and relations.)

Meta-database

Domain table

$TYPE_0$	E	TYPE OF SPORT	...	2	ACT_0	REC_0	...	T
$YEAR_0$	E	YEAR RECORD SET	...	1	REC_0	...	T_S	
$'NOO_0$	S	H	NUM_0					
NUM_0	E	STUDENT CARD NUMBER	NOO_0	STU_0	...	1	ACT_0	...	T_S
STA_0	E	STADIUM	...	1	ACT_0	...	T	
STU_0	S	STUDCARD	NUM_0					
$SURN_0$	E	STUDENT SURNAME	FIO_0	...	2	ACT_0	REC_0	...	T
FIO_0	S	FIO	$SURN_0$					

ACT_0	E	SPORTING ACTIVITY	4	NUM_0	$SURN_0$	$TYPE_0$	STA_0	...
REC_0	E	RECORDHOLDERS	4	$TYPE_0$	$YEAR_0$	$SURN_0$	NUM_0	...

Text value tables

$TYPE_0$

SKA_0	E	SKATING	
EQU_0	E	EQUEST. SPORT	EQS_0
EQS_0	S	ES	EQU_0
TRA_0	E	TRACK AND FIELD	
SKI_0	E	SKIING	

STA_0

DAU_0	E	DAUGAUA	
OIN_0	E	DINAMO	
IPP_0	E	IPPODROM	

$SURN_0$

IUA_0	E	IVANENKO	
IUA_1	E	IVANOV	
KON_0	E	KONNIKOVA	
SID_0	E	SIDOROV	
STA_0	S	STAROSTA	$YANS_0$
$YANS_0$	E	YANSON	STA_0

Database (user relations)

ACT_0

7901	IUA_1	TRA_0	DIN_0
7902	IUA_0	NULL	NULL
7903	SIO_0	SKI_0	DAU_0
7904	IUA_0	SKA_0	DAU_0
7905	$YANS_0$	EQU_0	IPP_0

. . . . . .

REC_0

SKA_0	1976	IUA_0	7615
EQU_0	1979	$YANC_0$	7905
TRA_0	1970	SKA_0	7047
SKI_0	1973	IUA_0	7328

. . . . . .

It is obvious that the architecture described here is not bound to an actual physical organization of records on direct access devices. It is only required that access to records be possible sequentially as well as by key.

The software implementation of this hybrid relational DBMS [3-5] uses the architecture described herein. Data are physically stored in a "dynamic page space" [3], organized on the B tree principle. This kind of physical organization is currently recognized to be very convenient and efficient [10].

The approach described here, providing a uniform organization of all relational DBMS meta-database system tables and making allowances for data semantics, is probably a new one; this completes a non-trivial development stage for relational DBMS. The meta-database architecture discussed here is for the most part implemented in PL/1.

REFERENCES

1. The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, Ed. by Tsichritzis and Klug, University of Toronto, Toronto, Canada. Information Systems. 1978, 3, p. 173.

2. Hammer M., McLeod D., "A Framework for Data Base Semantic Integrity" -- in: Second International Conference on Software Engineering, San Francisco, 1976, p. 498.

3. Tovstoles A. N., Kaplan E. L., Sheinkmar G. A. "A Logical Processor for External Storage for a Relational Hybrid DBMS", in the book A Hybrid Relational Database Management System. Conceptions and Implementation Methods. Riga. The Latvian Scientific-Technical Institute for Scientific-Technical Information [LatNIINTI], 1978, p. 25.

4. Kilov Kh. I., Popova I. A. "Inputting Data into a Data Base (DI-processor)", in the book A Hybrid Relational Database Management System. Conceptions and Implementation Methods. Riga. LatNIINTI, 1978, p. 19.

5. Kilov Kh. I., Popova I. A. "Logical Text Organization as a Part of Relational Database Architecture", in the book A Hybrid Relational Database Management System. Conceptions and Implementation Methods. Riga. LatNIINTI, 1978, p. 25.

6. Belonozov G. G., Novoselov A. I. "On Information Representation in Computer Storage". Automation and Computational Technology. Riga. Zinatie, 1975, No. 2, p. 65.

7. Won Kim. "Relational Database Systems" Computing Surveys, 1979, 11, No. 3, p. 185.

8. Winterbottom N., Sharman G. C. II. NDB: Non-programmer Data Base Facility. IBM UK Laboratories Ltd. Technical Report TR.12.179. September 1979, p. 55.

9. McLeod D. "High Level Definition of Abstract Domains in a Relational Data Base System". Computer Languages, 1977, 2, p. 61.

10. Comer D. "The Ubiquitous B-tree". Computing Surveys, 1979, 11, No. 2, p. 121.

Submittec for publication July 25, 1979.