Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

A DATA VALIDATION LANGUAGE*

PLAMEN B. DOUMKOV

A language especially designed for the description of data validation is discussed. The language is based on a tabular description of the validation procedures which the user would like to invoke in correspondence with his needs for checking the consistency of data. Four levels of validation are treated — single data item validation; checking of relationships between data items in one document; checking the consistency of data in batches of documents; and checking the integrity of data in a file or in a whole data base. The item validation is treated in some detail. Methods for defining users' data types are outlined.

1. Introduction. One of the current topics in the research activities devoted to problems connected with the gathering and processing of large amounts of data is the integrity and consistency of the data. The degree of reliability of the computer output information strongly depends on the solution of this problem. The need for satistfying consistency constraints and thus guaranteeing the correctness of data is being put forward recently as one of the preliminaries for the normal functioning of a data base management system [3, 4]. It is widely accepted that the data consistency can be achieved by a symbiosis of organizational and programming tools. A general treatment of the problem of data validation based on a comparison of the approaches taken generally toward the process of data and program input is presented in [1]. It is argued that some general programming tools are needed in order to perform syntactical and semantical data validation similar to the syntactical and semantical analysis inherent to the process of program compilation. The general structure of a data validation language was outlined in [1]. For satisfying all the requirements of the users it has been proposed that the basic way for description of the validation process should be in the form of tables.

For some very peculiar types of validation a procedural specification could be used. Such a combined language can be regarded as a powerful tool which could contribute to the solution and investigation of the problems of data consistency and reliability. Based on this proposal detailed specifications concerning the tabular part of the validation language were developed. This paper is devoted to a brief description of the language.

Several levels of data structures and data relationships subject to validation are identified:

— data item (field);
— relationships among items in a document;
— data relationships in batches of documents;
— consistency of data in a file or a data base.

* Delivered at the Conference on Systems for Information Servicing of Professionally Linked Computer Users, May 23—29 1977, Varna.

SERDICA Bulgaricae mathematicae publicationes. Vol. 4, 1978, p. 209—213.

2. Brief description of the data validation language. The language presented here gives to the user the possibility to describe easily and conveniently the validation needed for his data. For each of the four levels of validation a special specification sheet is provided. The sheets are in the form of tables and even non-programmers will have no difficulty to fill-in the sheets.

There is an additional specification sheet for defining users' data item types. A provision is made for a free form specification of validation procedures that cannot be easily described on the table sheets.

2.1. Data item validation. The data items are recorded into the fields of the documents designed for them. Each field can hold a fixed number of characters. Several fields form a physical record. The sheet "Fields Validation" (Fig. 1) is used for describing the data item validation.

The first three columns of the sheet are used to identify the fields where the data items are held and for describing their positions within the records. The next column is intended for specifying the need of allignment checking of the data item in the field.

The next column is used for description of the type of the data items. The system uses this description (possibly modified as shown below) for validating the data in the specified field by means of interpretative routines generated by each description. The approach usually used toward the description of the type, format and range of data is based on the tools available in the programming languages. For example the validation of the data in the fields AGE, BIRTHPLACE and SALARY could be performed according to their description

IDENTIFIER	FROM	TO	A	DATA TYPE	S	DATA TYPE MODIFICATION	
AGE	31	32	R	INT			
BIRTHPLC	33	49	L	CHAR			1
SALARY	51	60		DEC (7,2)			
AGE	31	32		DI2	R	(18;35)	
BIRTHPLC	33	49	L	LC5-16			
SALARY	51	60	R	DF7,2	R	(80,00;499,99)	2
BIRTHDATE	61	70		ESDATE			
WEEKDAY	71	80		WDAY	E	&DAILY&	
WEEKDAY	71	80		WDAY	N	&SUNDAY&	3
PARTNMB	1	8		U1			
DATE	11	20		U2			4
IDENTIFIER	1	10		ID			
FROM	11	13		DI1-3			
ALLIGN	17	17			E	&R&L& &	5
DATATYPE	18	24		TY	C	DEFINED BY SPECIAL PROCEDURE	

Fig. 1. Fields Validation specification sheet

in part 1 of Fig. 1. However, the data types shown in these examples do not reflect adequately the intrinsic character of the data inherent to the names of the data items. They are rather influenced by the operations that a computer is capable to perform and the consequently permitted data types in the procedural programming languages. A more precise specification of the data type is possible by specifying the expected data format in I/O operations. In reality the variety of data types is by far greater.

More generally, the concept of data item type can be considered as the set of those character strings of a given alphabet that correspond to the sense inherent to the name of the field where the item is recorded [1]. In order to ensure different levels of approximation for the set of meaningful values the user should be provided with some flexible techniques that could permit him to present a more exact interpretation of the data type.

Notation	Interpretation
DD	strings of decimal digits
LL	strings of Latin letters
CC	strings of Cyrillic letters
CL	strings of letters (mixed Cyrillic and Latin)
CH	strings of any characters
SC	strings of special characters only
DI	decimal integers
DF	decimal numbers with fixed point
BS	binary strings
HM	timing of the day in hours and minutes
HMS	timing of the day in hours, minutes and seconds
ASDATE	date in American Standard
ESDATE	date in European Standard
ADDRESS	address, street, number and place

Fig. 2. Sample data item types

The first step in this direction could be made by narrowing the meaning of the widely used in programming languages data types. The rather general, data type known as CHARACTER comprises a series of data types such as letter strings (they themselves could be of several different types — Latin, Cyrillic, Greek, etc.), mixed strings of letters and digits or letters and special characters and so on. It might be appropriate to consider also such data types as DATE, ADDRESS, TIME, etc. A partial list of such types is provided as Fig. 2. Part 2 of Fig. 1 shows trivial examples where several of these types are used for description of the validation of certain data items. The meaning of the data types is narrowed by specifying their format and possibly the range(s) of permitted values. The data type can be defined by explicit specification of the strings of characters which form the set corresponding to that type or by combining such explicit presentation with another type.

The format of the data item is described together with the specification of the type in the column "Data Type". The ranges of the values of the data items, explicit specification of some values and other modifications of the data type are possible by filling in appropriately the column designated as "Data Type Modification". The kind of the modification is entered in the one charac-

UNION			RESTRICION				CONCATENATION		NEW TYPE
			BY TYPE		BY RANGE				
UNION1	UNION2	UNION3	RESTR1	RESTR2	GE	LE	CONCAT1	CONCAT2	
DI3 DI2 ASDATE LL1	ESDATE				68	78	LL4 DL1-7	X2	U1 X2 U2 ID

Fig. 3. Types definition specification sheet

ter column, denoted by "S" on the sheet "Fields Validation". For instance R is for ranges, E is for explicit, etc.

If the user wants to describe validation of data types that are not available as standard built-in descriptions, he has at his disposal another powerful tool for specifying more exactly the sense inherent to his data type. He can make use of some operations on data types, the types being regarded as sets of strings. These operations, permit quite flexible data type definition.

A special specification sheet "Types Definition" is used for this purpose. The operations available are union of data types, restriction of one type by other types, concatenation of strings from different types. In addition restriction in the ranges can also be used in this sheet for convenience of the user. The union of two or more data types is a data type the strings of which belong to at least one of the data types participating in the union. The restriction is such a data type in which the strings are necessarily of the first type but they cannot be of the second (the restricting) type or types. As a result of the application of these operations new data types are defined that are designated on the sheet "Types Definition" (see U1, U2 and X2 on Fig. 3) and that may be used as type notation both on the sheets "Fields Validation" and "Type Definition".

2. 2. Document validation. The term document is a reflection of the approach taken in [1] toward the validation problem. The end goal of the validation is to discover errors in the source documents from which the data are extracted. A document in this treatment is a concept of the same range as a CODASYL set, table [2] or any other similar concepts. A document is a collection of several record types each having one or more occurences. The fields of each record type are described and named on several sheets "Fields Validation". The sheets "Document Validation" allow for description of the validation of relationships between different fields in the same record or fields in different record types but in one document. The language permits validation of the following classes of relationships:

— binary relations on data items;
— n-ary relations on data items;
— n-ary relations on binary relations.

Conditional validation steps can be governed by the use of indicators. The examples on Fig. 4 are self-explanatory.

2.3. Validating data consistency in batches of documents, whole files or data bases. Two more specification sheets are provided for the description of validation procedures needed to ensure data consistency. These are named "Batch Concisteny Validation" and "File Description". They

CONDITIONS	N-ARY RELATIONS	ARGUMENTS AND BINARY RELATIONS			INDICATORS ON	INDICATORS OFF
		A+B+C	=	D		
		UNITPRICE+UNITSOLD	=	TOTAL		
		BIRTHYEAR+18	<	MARRIAGEYEAR		
		MARRITALSTATUS	=	'SINGLE'	17	
17		NUMBERCHILDREN	=	0		
		MAX(S(I))	≦	300		
		KEY(I)	<	KEY(I+1)		
	ATLEAST2	SKILL='HIGH', AGE>	35,	EXPERIENCE>10		

Fig. 4. Document validation specification sheet

permit checking the number ot documents in a batch, validating proof totals. checking the ordering of the documents, tracing the completeness of a file or the integrity of a data base. The sheet "File Description" is used also for the description of the outlay and organization of the input and output files.

3. Experimental implementation and future development. Several basic components of the validation system have been experimentally implemented. The implementation is based on an approach of preliminary compilation of the validation description into an intermediate language and consequent interpretation of the intermediate language. This permits rather flexible modification of the system and allows for embedding of the validation sublanguage into a system for information servicing. This could be achieved by some changes of the compiler that would not affect the intepretative system. More thorough study of the problem of data integrity of the information base is being considered, possibly using some of the progress made by the use of the relational approach.

Acknowledgements. The author would like to express his thanks to P. Barnev for the support given throughout the work on the project and to K. Ivanov and A. Radenski for helpful discussions.

REFERENCES

1. P. H. Barnev, P. B. Doumkov. Analyzing the correctness of data prior to processing. *Serdica*, **2**, 1976, 350—359.
2. P. Barnev. Systems for information servicing of professionally linked computer users. *Serdica*, **4**, 1978, 164—179.
3. M. H. H. Huits. Requirements for languages in data base systems. IFIP Working Conference of Data Description Languages, 1975. Amsterdam, 1976.
4. M. Stonebraker. Implementation of integrity constraints and views by query modification. Proceedings of ACM SIGMOD Conference on Management of Data, San Jose, California, May 14—19, 1975.

Centre for Mathematics and Mechanics
1090 Sofia P. O. Box 373

Received 11. 9. 1977