Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

A SOFTWARE PROJECT WITH DATA BASE*

I. HAVEL, I. HOJDAR, P. LIEBL

The authors currently work on an operating system for a small computer that will be put into use in early 1979. The system will contain a data base and provide for multiprogram operation. The paper briefly states the theoretical points of departure of the project and the decisions adopted.

1. The necessity to conceive and build information servicing systems for groups of computer users who work on related problems and share data, leads to the introduction of the idea, and tackling of the problems of the Data Base (DB). Under DB we understand, following e. g. [1], the whole amount of (operational) data, describing a certain section of the real world (a good example is an industrial enterprise) stored in one computer (or in a network of computers) and shared by several users for the solutions of their respective tasks. The most satisfactory description of a DB we consider to be the r e l a t i o n a l m o d e l. According to this way of looking at the nature and structure of the data, a DB consists of a finite number of relations.

2. Given the nonempty sets $D_1, D_2, \ldots, D_p$ (the domains), a subset of their cartesian product is called a r e l a t i o n. The relational model of DC is based on the observation that an entity of the real world can be described by the values of several of its properties, a t t r i b u t e s, that is, represented by a r e c o r d which is an ordered p-tuple of values $d_1, d_2, \ldots, d_p$, where $d_i D_i, i=1, \ldots, p$. Each domain, the set of possible values of an attribute, is given a name referring to this attribute. As entities naturally group themselves into homogeneous classes of entities of the same t y p e, the set of records representing entities of one type forms a relation. In the real world, entities are interrelated in various ways. The relational model considers any r e l a t i o n s h i p between entities again as an entity and represents the connection between entity types as a relation.

3. The data in a DB are not only for inspection. They must be able to be u p d a t e d to reflect flexibly the changing facts about our section of the real world. There are 3 basic types of update operations: update proper (changing the value of one or several attributes in one record), amend (adding a new record to a relation) and delete (removing a certain record from a relation). The effect of update operations on the relation is that in fact we have a sequence of relations. We well say, not very accurately, that the relations are t i m e - d e p e n d e n t; the DB consists of a finite number of time-dependent relations.

---

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

SERDICA Bulgaricae mathematicae publicationes. Vol. 10, 1984, p. 128—131.

4. We will assume the relations to be finite subsets of the (not necessarily finite) cartesian products. A finite relation is conveniently represented by a list of its records; this can be understood as a two-dimensional table. Each column represents an attribute and each row is a record. On the intersection of a row and a column stands the value of the respective attribute of the entity represented by the respective record. As in a list its elements are naturally ordered, we take a somewhat modified view and consider the DB to consist of a finite number of (still of course time-dependent) ordered data sets (ODS). An ODS we define as a mapping of a section $(1, 2, \ldots, N)$ of the set of naturals into a relation; in other words, it is the numbered (and so, ordered) family of records of a relation. This modification of the standard definition enables us to introduce in an exact and convenient way notions such as "sorted" or "pointer". Moreover, it forms the basis for the introduction of a sublanguage for reading and updating which can deal with single records as well as with whole relations.

5. For the data organized in a DB it is a natural requirement to be accessible to more than one application program — otherwise they are the internal operational data of a program not deserving the name DB. From this it follows that on the computer (or computer network) where the DB is stored, the operating system (OS) provides for multiprogram operation. The means for describing such an OS and its function are programming languages for parallel programmming. We use Concurrent Pascal [2]. The basic notions of parallel programming are the generalized data structures called monitors and the concurrently running processes which share common data via the monitors.

6. The OS permitting concurrent execution of several application programs on a computer (or computer network) with a DB, together with all software, is represented by one single program in Concurrent Pascal. The parts of this program are, besides certain processes and monitors dealing with input/output processes each representing one application program and monitors (the DB-monitors) that contain all the data of the DB and organize access to them for the application programs. The compiler must provide for separate translation of (additional) processes (application programs) and DB-monitors as well as for their ad hoc linkage into the running program.

7. In a program in Concurrent Pascal, every process is syntactically almost similar to a program in ordinary Pascal [3]. The task of the application programmer therefore will be, in order to solve his problem, to write one or several ordinary sequential programs in a somewhat modified and simplified version of Pascal (e. g. no set, file or pointer type, different standard functions). To perform 1/0 operations, he will call procedures that are, from his view, standard procedures, but in the respective process of Concurrent Pascal which represents his program in the whole system, they are ordinary procedures that in turn use certain monitors. In a rather similar way, to perform read or operations with the DB, he will call standard procedures that "behind his back" use DB-monitors.

8. Let us give one example for the basic means the programmer will be able to use for communication with the DB. The statement employee (find, i, condition, buffer) has the form of a procedure statement with 4 parameters. The first is of type action = (find, get), the second of type integrer, the third is a function condition (b: employee type): boolean, and the

fourth as well as the single parameter of condition are both of type employee type = record name: string; born: record day: 1 .. 31; month: 1 .. 12; year: 1 .. 1977 end; department: 1 .. 20 end which can be considered to be the type of one record of the relation employee from the view of this program. The semantics of employee (find, i, condition, buffer) is as follows: In the relation employee a record is found for which condition equals true. The search starts with index equal i and after the successful search the index of the found record appears in i while the found record itself appears in buffer. If there is no record of the specified properties a value $i = 0$ is returned. The function condition is written by the application programmer. It may naturally contain in its turn operations with other relations or with employee itselt.

9. We consider data security [4] to be of considerable importance in a shared DB. By data security we understand the means at disposal to prevent unauthorized reading and updating. It is convenient to distinguish value independent security (certain attributes in a relation, or a whole relation, are protected) and value dependent security (certain records in a relation are protected). We speak respectively of column protection and row protection. While row protection can be implemented only dynamically, column protection can be, and should be, implemented statically. We propose to implement column protection, using inherent properties of the language Pascal. The programmer can use only the attributes explicitly stated in the type declaration (refer to the example given in 9.) type employee type, namely buffer. name, buffer. born, buffer. department. He has no way of using the fact, even when he learns about it, that in the DB the relation employee has also the attributes salary: integer and previously convicted: boolean. The "filtering" is done in the procedure employee which is not written by the programmer but supplied by the DB administrator at compilation time. Another application programmer (or, for that matter, the same one when writing a program for another department) might be issued with a standard procedure employee2, where now the type employee type 2 contains the attributes salary and previously convicted and the type action2 = (find, get, put, amend, delete).

10. As already mentioned, there is a DB administrator, a single person or whole department, who is in a certain sense in charge of the DB. In our view, he performs the following functions:

A) he decides on establishing a new relation in the DB, decides about its size (maximal value of index) and organization, about the type of the attributes, and writes the corresponding monitors and procedures.

B) he decides about access permissions to application programs (security) and writes the corresponding procedures.

C) he decides about the acceptance of application programs into the system and about their priority within the multiprogram regime and takes the steps necessary to compile and link.

Note that he has no immediate access to the data as such.

11. From what has been said it is obvious what we understand by "user" and "access to DB": it is the author of an application program and the statements in his program concerning the DB, respectively. The "casual user" as well as a non-procedural query language like SEQUEL are outside the scope. We consider including an interpretation system dealing with non-parametrical queries somewhere in the future.

12. All our considerations are based on a few preconditions. First, the computer must have disk memories for the DB. Second, the computer communicates with the outside world via terminals with interrupt facilities rather than via traditional 1/0 units that handle files in batch processing. Third, the system works in an environment where the values of the data are all defined (at least almost always) and are also (almost always) in due time supplied to the DB; in other words, the system may assume order and discipline as granted, rather than having to enforce both.

13. On the principles and ideas, the authors have suggested and are currently implementing in Czechoslovakia an OS with DB on the system KA10 (up to 128 kB direct access memory with 800 ns base cycle, several disk units of 1.25 MB each, four interrupt levels. rich and varied network of 1/0 devices of terminal type for collection and transmission of data) [5].

REFERENCES

1. C. J. Date. An Introduction to Database Systems. New York, 1976.
2. P. Brinch Hansen. The Programming Language Concurrent Pascal. *IEEE Trans. Software Engineering*, **1**, 1975, No. 2.
3. K. Jensen, N. Wirth. Pascal — User Manual and Report. *Lecture Notes in Computer Science*, No. 18.
4. H. Wedekind. Datensicherheit in Datenbanksystemen. In: Data Base Systems. *Lecture Notes in Computer Science*, No. 39.
5. KA10. Systém pro sběr a předzpracování dat. Praha, 1976

Matematicky ustav CSAV
Praha, Czechoslovakia

Received 11. 9. 1977