NEW REQUIREMENTS FOR USER INTERACTION WITH CAD/CAM DATABASES

F. Vernadat
Division of Electrical Engineering
National Research Council of Canada
Ottawa, Ontario K1A 0R8

RESUMÉ: Il est aujourd'hui communément admis que le système de base de données est un élément fondamental de l'architecture des systèmes de CFAO (Conception et Fabrication Assistées par Ordinateur). Cependant, les bases de données techniques actuelles sont fréquemment utilisées comme de simples structures de stockage de données pour alimenter des programmes d'application faits sur mesure. Il faut les améliorer, surtout dans leur interaction avec les utilisateurs, si on veut réaliser de vrais systèmes d'information technique pour la CFAO de demain. De plus, le système sera d'autant mieux accepté par ses utilisateurs si ses possibilités de communication s'avèrent simples, amicales et efficaces.

Dans le but d'accomplir ces objectifs, divers moyens d'interaction avec la base de données doivent être reconsidérés. Dans cette communication, nous discutons tour à tour les besoins en nouveaux types de données, en nouveaux modèles de données, en nouveaux langages de données, en nouveaux outils pour la maintenance d'intégrité des données et la gestion de la concurrence des données, et en nouveaux interfaces.

ABSTRACT: Today, it is commonly admitted that the database system is a fundamental element of a CAD/CAM system architecture. However, current engineering databases are frequently used as simple structured data repository devices to feed application tailored programs. They need improvements, especially in their interaction with users, to become real engineering information systems for tomorrow's CAD/CAM. Also, the whole system will be much more acceptable to its users if its communication facilities are simple, friendly, and efficient.

In order to accomplish these goals, several means of interaction with the database must be reconsidered. In this paper, we successively discuss the needs for new data types, new data models, new data languages, new tools for data integrity maintenance and concurrency management, and new interface facilities.

KEYWORDS: CAD/CAM, database management systems, interaction, data types, data models, data languages, interface.

INTRODUCTION

A central problem in the design of CAD/CAM systems concerns the database since it appears to be the backbone of CAD/CAM packages. The database is the means to link users and activities of the design and manufacturing process [32]. It must contain a model and all specifications of the object being produced from its early inceptions to its detailed drawings, process plans, NC programs, bills of material, and all information necessary to manufacture it. Databases have greatly extended the possibilities of CAD/CAM systems in many ways (central control of data, data sharing facility, independence of data from application programs, report generation facility,...) They are widely used in CAD/CAM application fields: mechanical engineering [13, 15,18,31], building design [2,9,27], VLSI [17,22], process control [19], machining [28,29], manufacturing [6], computer graphics [3,25,35], ... A recent, very comprehensive state-of-the-art discussion of the topic has been given by C.M. Eastman [10]. We refer the reader to this document considered as a work basis for general presentation of technical databases.

A recent bibliographic study has been completed at NRC to evaluate the global effort put into CAD/CAM databases

during the last twelve years [33]. The main conclusions of this work are:

- great emphasis has been placed on CAD databases, and very little on CAM databases
- database concepts are sometimes misunderstood, especially in CAM
- few packages are based on the state-of-the-art of database techniques
- developed systems are generally application-tailored packages
- most systems are centralized databases
- after a large usage of the CODASYL model, there is currently a rush on the relational model
- need for new data types
- need for new data models
- need for more sophisticated data languages
- need for tools to express, maintain, and check data integrity and data consistency
- the current evolution is toward distributed systems,
- and need for more efficient systems.

More efficient systems mean databases which provide the users with an interactive, friendly, cooperative, and versatile environment. Now the goal is to create such a system, and this is the reason why user interaction with CAD/CAM databases has to be reconsidered and improved.

INTERACTION CONTEXT

A CAD/CAM system is overall a data processing system. Various data are entered into the system, then they are processed to create new data using previously stored data to produce all data needed to manufacture a product. All this process is controlled and monitored by operators who need to communicate with the system and who need data to take decisions.

1. The users:

In the scope of a large system, conceived to support all the design and manufacturing activities or functions, we can distinguish three classes of users:

a. The application programmers: These are people who build programs which will process the data stored in the database. They are highly skilled programmers and they must deal with the data structures and the data schema. For example, they build application programs for finite element analysis in mechanical or civil engineering, simulation or routing programs for VLSI circuits, etc., depending on their fields of expertise.

b. The designers and application engineers: These users design objects and use the application programs interactively. Designers must be able to create, retrieve, update, or delete information. Through the system they interact directly with the content of the database, as do process planners or NC programmers to prepare the manufacturing of a part. In some cases, these users need to communicate with each other.

c. The end users: These people can only query the database for administrative or production management purposes (for example, to get a timely report on labor hours, materials on-hand, ...), or they can directly add or alter information in the database (add new machine-tool description, change constant data such as feed and speed rates, ...).

These users must be provided with a means to interact with the database system. Depending on how the CAD/CAM package is structured, the user directly accesses the DBMS or accesses it via the CAD/CAM system monitor.

2. Types of interaction:

Two types can be distinguished:

a. Logical interaction: This refers to means to alter the definitions, structures, and values of data. It concerns interaction with the content of the database. Available tools to perform this task are:

- the data types: they are used to define the data
- the data model: it is a model to structure the data and the relationships among data
- the data languages:
  - data definition language (DDL): it is used to declare the data types and the data model to the system
  - data manipulation language (DML): it makes possible the retrieval of pieces of information from the structure defined with the DDL
  - query language (QL): it provides the means to query the database content interactively
- tools to express data integrity and data consistency: they ensure that values of data are plausible or coherent between them.

b. Physical interaction: The interaction with the database system can be realized by physical interfaces such as keyboards, graphics interfaces (color terminals,...) sensitive screens, interactive plotters, voice input, ...

Obviously, interface requirements vary with the user type. This point will be expanded upon later.

3. Interactive, friendly, cooperative, and versatile environment:

People want to create real information systems for engineering purposes. This largely encompasses the notion of CAD/CAM equipment we know now. It means the integration of CAD/CAM possibilities with database and communication facilities. The database is no longer simply a data storage and retrieval system, but it must be a powerful (perhaps intelligent) tool to manage data of the object being designed and also data of previous products, technical data (from handbooks, catalogues, manufacturer labs, ...), estimation data (time and cost), etc. It must contain the know-how of the company and be able to communicate with other equipment. It must provide the users with an environment having the following properties:

Interactivity: By nature, design is an iterative and tentative process. Thus, interactivity is of prime importance. This facility can be implemented through a dialogue mode (questions and answers), by means of menus, or by means of command languages. For parts manufacturing, process planners and NC programmers also need to communicate interactively with the computer.

Friendliness: Many commodities must be implemented to make system use pleasant. In many software packages, when an error occurs the operator gets an error message (easily understandable or not) and the system stops. Also, the replies of the computer are often abrupt. These kinds of situations must be avoided. Data and messages must be clearly presented using pictures, drawings, sound, and combinations of these media in addition to traditional alphanumeric text.

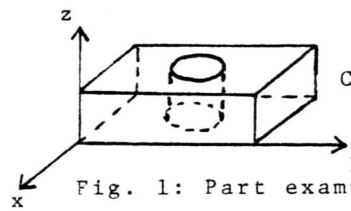
Cooperation: The interpretation of the information content of a database may vary from one user to another. Moreover, it is difficult to be sure that the given questions, especially for complex queries, will lead to the information desired. Thus, the system should be capable of answering the queries, but it should also give additional data closely related to the questions and perhaps suggest data of interest. Use of multi-media terminals (with color facilities, sensitive screens, audio input or output) can provide a more cooperative environment. Finally, naive users, or non-computer-specialist users must be provided with tools to help them to discover and to browse through the structure and the content of the database with no keyboard device, as much as possible.

Versatility: In current database systems only unambiguous questions can be answered, and to express the query, the user has to know precisely the conceptual model of the data stored in the database. However, in the CAD/CAM context, a very common situation is the need for a user to consult old versions of a product, which may have been designed by another person, and it becomes difficult to clearly formulate the questions. The system must allow examination of the schema of the data, and help the user to pose the questions. Eventually, in cases of ambiguous questions, it can reformulate the query for each alternative and ask the user which is the best.

In order to achieve this goal, which may appear Utopian to some, we think it necessary to reconsider all the points mentioned in the paragraph on types of interaction. This is the aim of the following section.

RECONSIDERING USER INTERACTION WITH CAD/CAM DATABASES

Suppose we want to describe a square part with a hole in it.



Part designation:
part name = shim
part code = S21

Comment: shim 21 is to be used with part E232

Fig. 1: Part example

Constraint: The axis of the hole is at the middle and is perpendicular to the upper face
Attached message: Add a wedge end to the right end of the part

1. More data types are required:

Most DBMS packages only support real, integer, and character string data types.

Obviously, for our example, the database system must be able to simply and directly handle other types of data to store more information.

a. Geometric data: Since CAD has been confused for a long time with computer-aided drafting, geometric modelling has become an extensive field of research. There are three methods for modelling solid objects [1] and each has its own set of primitive structures:

- wire-frame modelling:
  primitives: points, lines, arcs, circles, curve splines
- the boundary representation:
  primitives: points, lines, arcs (circular, conic, cubic), circles, planes, splines, polyhedra
  A polyhedron is decomposed into vertices (points), edges (lines), and faces (planes)
- the constructive or volumetric representation:
  primitives: cubes, blocks, cylinders, wedges, segments, fillets, tetrahedra

The first approach is used in reference 25 for two-dimensional drawings, and the second method is used in reference 3 for solid objects. Both packages are interactive relational graphics databases for which the data types are described by fixed format relational tables.

b. Symbol data (or icon data): Most engineering fields use specific types of data which are represented by symbols. For example, in electronic design, AND-gates and OR-gates are frequently used:

Fig. 2: Symbol data examples

These data types can be declared by the following statements:

```
TYPE AND INPUT-PINS = A,B   OUTPUT-PIN = C
TYPE OR  INPUT-PINS = D,E   OUTPUT-PIN = F
```

c. Text data: To completely describe our example (figure 1), we need to include text in the model to take into account the drawing comments and eventual attached messages giving instructions. We distinguish

- non-formatted text data: it refers to text as it is usually written
- formatted text data: this is a text with a predefined format, for example, process plans or NC programs in CAM.

In both cases this information can be stored in a tabular form (or relation) having a name and two attributes: line number (this is an integer), and line information (this is a character string limited in size to the maximal number of characters per line).

d. Vectors and matrices: During the analysis step of the design stage, many mathematical calculations have to be performed, especially for finite element analysis, for instance. Very often, this requires the processing of vectors and matrices.

2. New data models are needed:

a. Classical and hybrid data models: Currently, most of the CAD/CAM database systems developed use the three classical data models: hierarchical, network, or relational approaches [7]. However, they begin to be criticized [5,20,34]. The hierarchical model can handle only one-to-one and one-to-many relationships among data. This is too restrictive. Common limitations of the hierarchical and network models are:

- their data schema is essentially static, while design necessitates dynamic schema since it is a tentative and iterative process
- the user must know all access paths to the data and has to deal with pointers, so they do not provide a clear distinction between the logical and the physical levels of data
- their implementation complexity which implies a complexity of the data languages

The relational model represents a major contribution in the field of data modelling, since it allows a real separation of the logical and physical levels of data. This ensures a real independence of data from application programs. The current emphasis on relational database design is largely due to the simplicity and the strong mathematical basis of this approach, permitting the implementation of high-level data languages. Another advantage of this concept is that it supports a dynamic schema because tables (or relations) can be split or aggregated or modified to give new tables, or they can be deleted. However, this model forces the user to think in terms of relations, which may be quite unnatural for some applications. Also, the description of multiple views of complex objects may induce the creation of many tables and provoke data redundancies.

A relational/network hybrid data model has been proposed [16] to take advantage of both approaches. Large tables are decomposed into smaller ones, and are linked along some semantic considerations in a network fashion. But, all of these solutions suffer from common weaknesses:

- they do not help to capture the semantic meaning of data
- they do not assist the user to express (semantic) constraints on the data
- they do not easily model many-to-many relationships

b. Semantic data models: This is an emerging class of models in database theory. The purpose is to develop higher level primitives able to handle more abstract data types and which directly represent many-to-many relationships. Syntactic basic elements of semantic data models are: the attributes, the entities, and the associations. Each of these has a type. This data type notion is the same as in programming languages [15].

Three main types of associations have been identified as bases in semantic data models [23]:

- the "has-subtype" association or "is a"
  example: a cube is a block
- the "has-attribute" association or "part-of"
  example: a point has coordinates
- the "has-occurrence" association or "has-member"
  example: CENTRE is a point, if CENTRE is the centre of a circle.

The quantity of basic elements of the semantic model makes the design of databases more difficult. In compensation the user is provided with a tool allowing him to create a more general model than with classical models. Some forms of modelling integrity can be ensured. More investigations remain to be done on this approach.

c. Attributed grammars: If we come back to our example (figure 1), it is possible to describe it with a semantic model. However, it is difficult to simply answer the questions:

Is there a hole on part shim S21?
Where is the hole on part shim S21?

One way to handle this is the use of attributed grammars as suggested by the constructive representation. Suppose P is the part; P can be decomposed into a block B and a cylinder C, and P is the difference (-) of B and C. (See fig. 3).

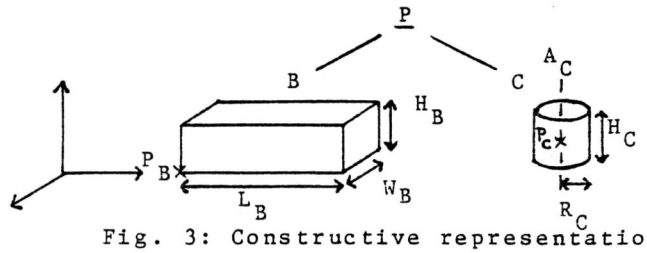


Fig. 3: Constructive representation

If A(X) is the list of attributes of element X, we have:

$A(B) = (P_B, L_B, W_B, H_B, +)$

$A(C) = (P_C, R_C, A_C, H_C, -)$

P_B, and P_C are points of reference to locate the element, A_C is an orientation axis (its type is line), + indicates if there is material, - indicates if there is not.

The grammar is defined by:

- a syntactic rule:
  $P \longrightarrow B\text{-}C$
- semantic rules (the hole is in the centre of B):

$$x_C = x_B - \frac{W_B}{2} \qquad \begin{cases} P_C\ (x_C, y_C, z_C) \\ P_B\ (x_B, y_B, z_B) \end{cases}$$

$$y_C = y_B + \frac{L_B}{2}$$

$$z_C = \frac{H_B}{2} \qquad H_C = H_B$$

Now it is easy to know if there is a hole in part P (we look for the sign - in the list of attributes of elements of P) and to find its location.

d. Some aspects of the object to design are sometimes difficult to model since they are not well defined or their properties are not precise (for example, an object has to be compact, which gives a fuzzy relation between its dimensions). Dubois [8] has carried out some investigations on the application of fuzzy sets theory to CAD.

3. Complete data languages:

a. The data definition language (DDL): The DDL is used to describe the structure of the logical database. A complete DDL is a language which can

- support the definition of complex data types
- provide basic primitives of description
- allow expression of consistency constraints

Definition of complex data types [4]: If we look at the example of figure 1, to describe the object we need point, block, and cylinder types. These can be easily defined with a language such as Pascal using record types:

```
Type point = record
        pointname: integer;
        xcoord, ycoord, zcoord: real end;

type block = record
        blockname: integer;
        refpoint : integer;
        blength,bwidth,bheight: real end;
```

Underlined expressions are reserved words.

Other complex types can be described:

- enumerated data types:
  ```
  type operations = (drilling, milling,
                     turning, boring,
                     tapping);
  ```
- structured data types:
  this permits declaration of vectors and matrices
  ```
  type vector = array [1..3] of real;
       strengthmatrix = array [1..3,
                        1..3] of real;
  ```

Basic primitives: Primitives of the DDL are useful to define the structure of the data in terms of the data schema. The description of the schema is usually stored in the data dictionary. In the case of a relational database, the basic primitives are:

DBNAME: to give a name to the data schema

DOMAIN: to declare the domains on which values of attributes are drawn with their type

example: DOMAIN OPE: operations

RELATION: to create a relation, indicating its attributes, their domains, and the key attribute(s)

example:

```
RELATION TRIANGLE (TRINO : integer;
                   PNT 1 : integer;
                   PNT 2 : integer;
                   PNT 3 : integer)

          KEY (TRINO)
```

END: to terminate the description session

Other primitives have to be provided to allow the description of different views of data.

Expression of consistency constraints:

Typing can be seen as a means to express integrity constraints on data, i.e., ensuring the data value is plausible. However, one has to be sure that data values are consistent among them. So, the DDL must supply the user with facilities to fix constraints on data. These are assertions about the content of the database and they must hold at all times. Examples of constraints:

integrity constraint: a distance is never negative.

consistency constraint: the diameter of the hole of shim S21 is smaller than the width of the part.

These are usually expressed by logic expressions with predicates. An example of such a sophisticated DDL is given in reference 20 for logic circuits.

b. The data manipulation language (DML): This language permits the manipulation of the database content or structure either by application programs or by users (designers, for instance). It consists of basic primitives. In the case of a relational model:

INSERT: to insert a tuple in a relation

DELETE: to destroy a relation or a tuple in a relation

MODIFY: to change the value of attribute(s) in a relation

GET TUPLE(iTH,NEXT,or BY KEY): to read information in a relation

GET RELATION: to read all the content of the relation

To complete the DML facilities, some systems include graphics manipulation primitives such as translation, rotation, scaling, union, difference, intersection, and correlation. However, one can advocate that these are more CAD language commands than database commands, since they are performed by means of dedicated algorithms. This would also be true if the DML included manipulation commands for text data (expand, compress, etc.).

c. The query languages (QL): They have been extensively studied in database theory, and quite complicated questions can be answered by the system. Very often they are embedded with the DML (see Query-By-Example). However, they are generally neglected in CAD packages since their use is not common in design and manufacturing. But, with the evolution of CAD/CAM databases towards engineering information systems, they will (and they begin to) be very useful for process planners, NC programmers, or people working in Production Management activities.

Currently the evolution is toward special-purpose languages designed for specialists of a given field (piping, VLSI design, etc.) and natural languages.

These languages are for non-computer specialists.

The common problems which arise with data languages are
- should the DDL, the DML, and the QL be integrated?
- what kind of implementation should be chosen -
  - menu commands?
  - interpreted languages?
  - compiled languages?

It has been advocated that the DML and the DDL must be embedded for use by designers since CAD is a trial-and-error process. However, this can lead to strong data inconsistencies and create conflicts with application programs which access these data if their structure is modified frequently. Obviously, this requirement varies with kinds of users: only designers need to access the DDL (maybe a database administrator, too), application programmers need only to know commands of the DML, engineers have to use the DML and the QL, and end-users essentially consult the database content with the QL.

Concerning the second question, having compiled DDL and DML makes it possible to use the notion of data types as it is defined in Pascal, for example, and to construct complex types. Also, compiling is an error detection facility and gives fast response time at execution. On the other hand, in a compiled database it is difficult to modify the data schema, which becomes essentially static, but overall the interactive environment created by use of interpreted commands or menus is lost. Nevertheless, it may be desirable to have a compiled database schema for application program coherence and standardization. So, the DDL can be compiled, and commands of the DML and QL can be interpreted and executed by subroutines to provide interactivity.

4. New tools for integrity, consistency, and concurrency control:

The previous paragraphs focussed on the problem of defining the schema (or structure) of the data. But once the data have been organized, they are entered or maintained in the database only if they are accurate and compatible in the application context. This is a difficult task called integrity and consistency maintenance and checking, which is especially hard to manage in engineering databases because concurrent and/or unfinished object representations are allowed to co-exist. Very often in current CAD/CAM DBMSs this feature is not discussed and implemented. The problem has been tackled with great emphasis only recently [10,11,12,21,24]. Due to the length limitation of this paper, let us only express that:
- data inconsistency can be avoided if data redundancy is not allowed. However, most engineering databases support several designers working on the same project at different sites and data duplication is sometimes unavoidable.
- the semantic integrity concerns the meaning of data with respect to the real world situation described. From the specification and description of a new product to its manufacturing and assembly statements, many data constraints have to be expressed, checked, and maintained. These are stated by means of semantic integrity constraints, and maintained by integrity transactions, i.e., a structure to manage the semantic integrity of the database without necessarily guaranteeing it all the time [11].
- global and local integrities can be distinguished. Global integrity concerns the whole database and may only be reached at the end of the design process. Local integrity concerns constraints on data values and can be checked when the data are entered or updated (e.g., the value must belong to a domain specified in the DDL). In reference 21 is specified the need of delayed integrity checking in order to avoid an explosion of data updates sometimes due to one data change. Integrity is checked after all data changes have occurred for a given task.

Another important topic is concurrency control. Since engineering databases operate in a multi-user environment, some conflicts may occur when data are accessed or updated. Generally, transaction locking mechanisms are used on READ or WRITE to solve these problems [12,26].

Some tools to manage integrity controls already exist. Nevertheless, much remains to be done, especially concerning global integrity maintenance, and the design of languages to express the integrity constraints.

5. New interfaces have to be designed:

They must:
- make data exchange easy
- give a clear presentation of data

- provide the means to get data

Graphics devices: These last years, graphics devices have been greatly improved in terms of color facility and resolution. They are widely used by designers who now need sensitive screens, audio-signals, and voice inputs to speed up data communication. They also use the zooming, shading, etc., possibilities and the light-pen facility. Moreover, designers, engineers, and end-users want to be able to display interactively the schema of the database content, and even to be able to query it without language and keyboard, but just with a menu and sensitive screen. This is necessary for non-computer specialists, for designers who do not want to waste time learning complex languages, or for naive users, i.e., people who do not know or do not remember the data schema (previous designs, ...).

Browsing facility: This refers to the possibility of consulting the database content in the same manner as we do when we search in a book. The data handled are here essentially numerical or textual. To realize this a powerful query/edit interface has to be conceived, including the possibility to locate a data-item with a light-pen and eventually to zoom the data.

Spatial data management: The organized representation of data on a screen, sometimes without languages and keyboard, is another interesting interface facility. It is convenient for users with no knowledge of DBMSs, no knowledge of the database content (so, it is not worthwhile to remember it), and it makes the browsing possible [44,30].

CONCLUSION

For several years the attention of researchers in the database field has been on data representation and structures. Currently their efforts are bearing on data meaning and behaviour. This is also true in the CAD/CAM field. Since in engineering, database systems will have to be used by a wide range of users it is important to consider the facilities for user interaction and to build them in the perspective of the future engineering information systems.

REFERENCES

1. Borgerson, B.R., and R.H. Johnson (1980). Beyond CAD to computer-aided engineering. In Information Processing 80 (S.H. Lavington, Ed.). North-Holland, Amsterdam. p. 659-66.
2. Borkin, H.J., J.F. McIntosh, P.G. McIntosh, and J.A. Turner (1982). ARCH-MODEL 1.3 Geometric Modeling Relational Database System. Architectural Research Lab., Univ. of Michigan (Ann Arbor, MI), June.
3. CAM-I (1979). Design of an Experimental Boundary Representation and Management System for Solid Objects. Tech. Rep. R-80-GM-02, CAM-I (Arlington, TX), Nov.
4. Challis, M.F. (1982). Typing in data base models. In File Structures and Data Bases for CAD (J. Encarnaçao, and F.L. Krause, Eds.). North-Holland, Amsterdam.
5. Cholvy, L., and J. Foisseau (1982). Nature, fonctions et modèles des bases de données dans un système de CAO. Proc. 2nd European Conf. on Computer-Aided Design in Small and Medium Size Industry MICAD 82 (Paris, F), Sept., 55-65.
6. Colding, B.N. (1981). Generating a Manufacturing Data Base - An Overview of Requirements. Tech. Paper MS81-181, Society of Manufacturing Engineers (Dearborn, MI), Apr.
7. Date, C.J. (1981). An Introduction to Database Systems, 3rd ed. Addison-Wesley, Reading, MA.
8. Dubois, D. (1980). L'apport possible de la théorie des ensembles flous à la CAO. Bulletin MICADO, No. 31, Feb.
9. Eastman, C.M. (1980). GLIDE 2 User's Manual. Inst. of Building Sciences, Carnegie-Mellon Univ. (Pittsburg, PA).
10. Eastman, C.M. (1981). Database facilities for engineering design. Proc. IEEE, 69 (10), 1249-63.
11. Eastman, C.M., and G.M.E. Lafue (1982). Semantic integrity transactions in design databases. In File Structures and Data Bases for CAD (J. Encarnaçao, and F.L. Krause, Eds.). North-Holland, Amsterdam.
12. Eastman, C.M., and A.R. Kutay (1982). Transactions and concurrency in engineering databases. Proc. TIMS-ORSA Conf. (Detroit, MI), Apr.
13. Fischer, W.E. (1979). PHIDAS - A database management system for CAD/CAM application software. Comput. Aided Des., 11(3), 146-50.
14. Friedell, M. (1982). A graphics interface to large, shared databases.

Proc. Graphics Interface '82 (Toronto, Ont.), May, 271-4.
15. Grabowski, H., and M. Eigner (1979). Semantic data model requirements and realization with a relational data structure. Comput. Aided Des., 11(3), 158-68.
16. Haynie, M.N. (1981). The relational/network hybrid data model for design automation databases. Proc. 18th Design Automation Conf. (Nashville, TN), June, 646-52.
17. Katz, R.H. (1982). A database approach for managing VLSI design data. Proc. 19th Design Automation Conf. (Las Vegas, NV), June, 274-82.
18. Kimura, F., Y. Yamaguchi, Y. Sasaki, K. Kido, and M. Hosaka (1982). Construction and uses of an engineering data base in design and manufacturing environments. In File Structures and Data Bases for CAD (J. Encarnaçao, and F.L. Krause, Eds.). North-Holland, Amsterdam.
19. Koller, H., and K. Frühauf (1981). A data base management system for industrial process control. Comput. Ind., 2, 171-7.
20. Lacroix, M., and A. Pirotte (1981). Data structures for CAD object description. Proc. 18th Design Automation Conf. (Nashville, TN), June, 653-9.
21. Lafue, G.M.E. (1982). Directed Integrity Dependencies and Performance of Delayed Integrity Checking. Computer Science Dept., Rutgers Univ. (New Brunswick, NJ).
22. Leyking, L.W. (1979). Database considerations for VLSI. Proc. Caltech Conf. on Very Large Scale Integration (C.L. Seitz, Ed.), Jan., 275-301.
23. McLeod, D., and J.M. Smith (1980). Abstraction in databases. Workshop on Data Abstraction, Databases, and Conceptual Modelling (Pingree Park, Co), June, 19-25.
24. Neumann, T., and C. Hornung (1982). Consistency and transactions in CAD databases. Proc. 8th Int. Conf. on Very Large Data Bases (Mexico-City, Mex.), Sept., 181-8.
25. Patnaik, L.M., and N. Ramesh (1982). Implementation of an interactive relational graphics database. Comput. and Graphics, 6(3), 93-6.
26. Rasdorf, W.J., and A.R. Kutay (1982). Maintenance of integrity during concurrent access in a building design database. Comput. Aided Des., 14(4), 201-7.
27. Spoonamore, J.H. (1982). CAEADS - Computer-Aided Engineering and Architectural Design System. Tech. Rep., U.S. Army, Construction Engineering Research Lab. (Champaign, IL), Aug.
28. Taraman, S.R. (1981). Machining Data Bank Structure. Tech. Paper MS81-490, Society of Manufacturing Engineers (Dearborn, MI).
29. Throop, J.W. (1981). A Common Data Base for Metal Cutting Data. Tech. Paper MS81-183, Society of Manufacturing Engineers (Dearborn, MI), Apr.
30. Tuori, M. (1982). Spatial information management - A survey. Proc. Graphics Interface '82 (Toronto, Ont.), 243-50.
31. Ulfsby, S., S. Meen, and J. Øian (1981). TORNADO: a DBMS for CAD/CAM systems. Comput. Aided Des., 13(4), 193-7.
32. Vernadat, F. (1983). Communication: a key requirement in computer integrated manufacturing. Proc. 2nd Annual Phoenix Conf. on Computers and Communication (Phoenix, AZ), March.
33. Vernadat, F. (1983). A Commented and Indexed Bibliography on Data Structure and Management in CAD/CAM: 1970-1983. Res. Rep. ERB-956, National Research Council (Ottawa, Ont.), May.
34. Vernadat, F. (1983). Database applications to CAD/CAM. Submitted to IFIP '83 - 9th World Computer Congress (Paris, F), Sept.
35. Weller, D., and F. Palermo (1979). Database requirements for graphics. Proc. Spring COMPCON 79 (San Francisco, CA), Feb., 231-4.