Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7441--94-7539

Digital Mapping, Charting, and Geodesy Analysis Program Technical Review of Vector Product Format (VPF)

H. VINCENT MILLER

Mississippi State University
Prepared for Marine Geosciences Division

S. DANETTE COUGHLAN
KEVIN B. SHAW
SUSAN V. CARTER

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

November 29, 1994

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
November 29, 1994

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Digital Mapping, Charting, and Geodesy Analysis Program Technical Review of Vector Product Format (VPF)

5. FUNDING NUMBERS

Job Order No.	5745137A5
Program Element No.	0603704N
Project No.	R1987
Task No.	300
Accession No.	DN257086

6. AUTHOR(S)
H. Vincent Miller*, S. Danette Coughlan, Kevin B. Shaw, and Susan V. Carter

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/MR/7441--94-7539

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Tactical Oceanographic Warfare Support Office
Stennis Space Center, MS 39529-5004

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
*Mississippi State University

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The Vector Product Format has been instrumental in providing the Defense Mapping Agency (DMA) with a standard format in which to store, port, and effectively maintain vector databases. As with practically any standard, however, current advances necessitate changes. NRL's Digital Mapping, Charting, and Geodesy Analysis Program (DMAP) has been given the opportunity to suggest changes which would enhance the Navy's use of VPF. DMAP's comments range from specific essential/suggested changes to more broad issues of data quality and topology. Suggested figures are given to clarify sections of text.

14. SUBJECT TERMS
requirements, MC&G data, mapping, DCW, ADRG, WVS

15. NUMBER OF PAGES
18

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Contents

Digital Mapping, Charting, and Geodesy Analysis Program
Technical Review of Vector Product Format (VPF)

1.0 Introduction

This review details recommendations for improving the Vector Product Format (VPF) draft standard [1]. VPF is a standard format, structure, and organization for large geographic relational databases. The Digital Nautical Chart and World Vector Shoreline are two prototypes of VPF databases. In 1991, VPF was evaluated by the Digital Mapping, Charting, and Geodesy Analysis Program (DMAP). That unpublished review (reprinted in Appendix B) took the form of a recommendation against the relational database format and toward an object-oriented format. Since VPF's current form is unlikely to change due to widespread use, DMAP has in this review provided comments to strengthen the relational format of VPF.

Although a definite enhancement over the previous version [2], this new version continues to have some of the same weaknesses as its predecessor. These errors and recommended corrections are enumerated in the lists of the following sections. In a more general sense, the VPF continues to be deficient in certain broad areas. For example, all VPF-compliant products should clearly state which of their tables are non-VPF specific. This stipulation, which would expedite product implementation, should be a requirement of the VPF standard itself. Another issue is Data Quality. More of the data quality tables need to have mandatory status, including lineage.

The extent of topology is another topic that should receive more attention in the VPF specification. According to [1], topology refers to any relationship between connected geometric primitives (e.g., points, lines, areas) not altered by continuous transformation. Some current VPF product specifications reviewed by DMAP state that topology is preserved only within a coverage. As defined by [1], a coverage is a set of feature classes (e.g., feature class *roads* is contained in the *transportation* coverage) where primitives interconnect as described by the prescribed topology. The fundamental issue of the extent of topology must be addressed in the VPF written standard as well as individual product specifications.

Finally, two implementation details should be discussed. First, the shape line, while referred to as hypothetical, should be labeled for future research if it has not been officially implemented. Second, the issue of physically representing tiles, while product dependent, needs attention. For example, some products (VMap Level 1 Prototype 2) use a directory for each letter in the GEOREF tiling scheme, creating a hierarchy such as f/j/h/b. Other products (DNC Product 1) may group this structure into one directory, such as /fjhb. Does one implementation have an advantage over the other? Are there other alternatives? Appendix D should give examples.

2.0 List of Essential and Suggested Comments

The following list supplies comments classified as "essential" or "suggested." Page numbers and line/figure/table positions are given, as well as recommended alternate text.

<u>KEY</u> P = page L = line
T = table F = figure

2.1 Essential

P 20 L 38 The file type *index* should be included with directory and table, both here and in TABLE 1.

P 22 L 16 The specification states that "a primitive table (discussed in section 5.2.2.1) may possess these two tables ... ," referring to an index file and a narrative table. Can primitive tables have associated narrative tables? They are not listed as optional in FIGURE 7.

P 24 L 4 Make a reference to "see TABLE 2" after the position of the narrative table name is given.

P 27 F 6 This FIGURE, together with FIGUREs 9 (p. 34), 16 (p. 43), and 17 (p. 45), should provide the VPF user with the most comprehensive visual overview of what files are available and their optional/mandatory status in VPF. However, the symbols do not effectively distinguish between table/directory/index. Moreover, some symbols in FIGURE 9 contain shadows (Does this indicate multiple copies?). These figures should be regenerated similar to Figure 1 of this review, which not only displays optional/mandatory status but also the file type (table/directory/index). Parallel traits (DHT and LHT) between structural levels are also more clearly visible. Note: In this figure, some VMap, WVS, and DNC product specifics are shown. Also, the primitives are displayed as optional to indicate that not *all* primitives are required in a given tile directory.

P 28 L 19 A reference to Appendix B should be included after "winged-edge topology," since this occurrence is the first.

P 41 L 18 In the previous version of the VPF specification [2], Section a. dealt with primitive/tile id columns vs. the triplet id in a feature table. This section is missing in the current specification. Is this construction no longer valid?

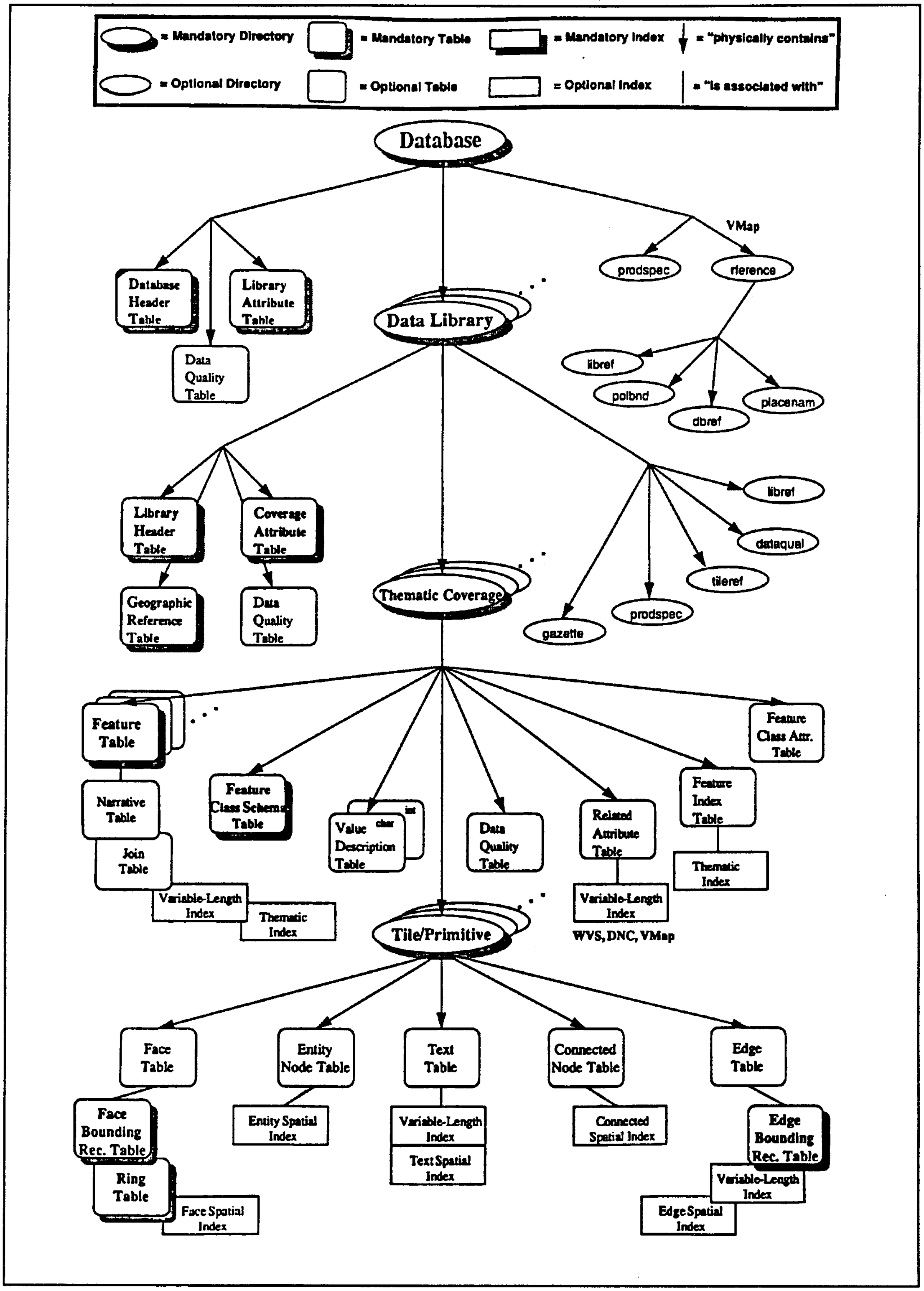

Figure 1. Suggested overview figure to show optional/mandatory status.

P 45 T 9 "Feature attribute table" should be "feature table." Also, the last two rows of this table should be grouped to show that together they represent the bottom structural level "Feature Class."

P 48 T 13 This TABLE contains an incorrect heading: "Column name" should be "op/man status" or something similar.

P 49 T 14 The following names most likely are reserved and should be listed: DQ*, where * represents AREA, LINE, POINT, or TXT.

P 49 T 15 *DATAQUAL* would probably be easier understood as just *DQ* (as it is in VMap). Also, *TITLEREF* should be changed to *TILEREF*.

P 49 T 16 The following suffix should be included: .fti, to indicate "Feature index table thematic index." Also, .rat (related attribute table) should be a reserved name.

P 51 T 17 The FIRST_EDGE column description should read as follows: "Always null (included for compatibility)." The Column Type for the Coordinate column is C/Z/B/Y. This description should be clarified (what does the "/" symbol mean?).

P 54 T 21 Specify "connected node" when describing the START_NODE and END_NODE.

P 55 L 14 "A ring within a ring contained within a face (e.g., a lake within an island which is contained within a larger lake) has no topologic relation to the outer face (the larger lake)." Is this sentence true, even if all are contained within the same coverage? Perhaps "topologic relation" is not being used properly in this context.

P 60 L 1 To be consistent with earlier definitions, the area, line, point, and text should be referred to as *simple*. The word *tables* in line 2 should most likely be *classes*.

P 61 T 32 The footnote to this table refers to a "feature class table name," which should be a "feature table name."

P 69 L 15 Should a database contain a data quality coverage, or any coverage for that matter? A database by definition is a collection of libraries, not coverages.

P 73 L 6 Again, should a coverage (in this case, *names placement*) be present in a database directory?

P 78 T 51	From a programmer's viewpoint, the bucket size should be recorded not only in the product specification but also in the header of a spatial index. The question arises: Will the bucket size be consistent for all spatial indexes within a tile? A coverage? A library?
P 78 T 51	*Node*, although correct in this context, is probably not a good name for the bins (cells) created in spatial indexing. *Bin* would be less confusing, since nodes are also primitives. Also, *tree* seems out of place in this part of the text. *Index* would suffice in its place.
P 78 L 32	The sentence should read: "This record is a two-dimensional array the length of which is NNODE described in the header record."
P 79 T 53	The footnote describes c incorrectly. The correct definition is as follows: c is {0 ... number of primitives for a node - 1}.
P 82 T 55a	The first integer value should be defined as "header length + index directory length" rather than just "header length." Also, two lines below this definition, the value definition should be "number of indexed rows."
P 119 F 33	A second pointer should be added from the tile 2 primitive to the thematic index.
P 123 F 37	How would a thematic index created on a primitive be named, as per the conventions on page 49? Thematic indexes are not listed as optional files in FIGURE 7.
P 132 L 1	This section 40.6.2 probably needs clarity on the type of performance.
P 135 L 1	This section 40.6.5 probably needs clarity on the type of performance.
P 140 L 30	Since there is an ever-increasing trend away from FACS for VPF products and toward DIGEST FACC, the reference should be eliminated.
P 146 F 53	The file name DQ.COM has an extension which is not in TABLE 16. If it represents "complex," the correct extension is .CFT.
P 153 L 8	The equations given here are clearly linear stretches between 0 and 255, with the exception of a "+ 1" added for the equations used to compute maximum coordinates. Is there a reason for this "+ 1" term other than the fact that points have essentially trivial bounding rectangles?
P 160 T 64	*Features* should be *primitives*.

P 160 T 64 The bin array record and the bin data record should be labeled.

P 161 L 41 Insert "Since face 13 does not include the query point" at the beginning of this sentence to indicate the reason to continue the algorithm.

2.2 Suggested

P 17 L 13 The phrase "from fully layered to completely integrated" is confusing. What is the definition of "completely integrated," and what is an example of such a data organization?

P 26 F 5 The change of the lower structural level from "Primitive/Feature" to "Feature Class" was a helpful one. The previous definition tended to blur the distinction between features and feature classes. However, now the specification content headings seem inconsistent with the structural levels. For example, "5.2.2.1 Primitives" is followed by "5.2.2.2 Feature classes" which is followed by "5.2.2.3 Coverage," etc. This is repeated again in section 5.3. An alternative would be to introduce primitives in introductory paragraphs.

P 36 F 11 This FIGURE, together with its companion FIGUREs 12 on page 37 and 13 on page 38, is an essential part of the specification and has improved since the previous VPF specification. Showing the columns and pointers within the tables would be a more natural approach. For example, Figure 2 of this review displays an alternate way of showing the levels of topology. The reader sees immediately which columns are used, which columns are optional or null, which are pointers, etc.

P 46 L 32 "Column data type," "Field Type," and "Column Type" all used here and in TABLES 10 and 11 to refer to the same entry. A standard name is needed, preferably field type.

P 51 T 17 The text describes the entity node primitive as being composed of three columns, whereas TABLE 17 shows four columns (one is null). The three mandatory columns should be more obvious. This type of comment could be applied to TABLES 19, 21, 23, and 27.

P 61 L 27 This sentence seems to contradict the footnote of TABLE 32, regarding field type K (triplet id).

P 68 L 12 "AFT" should be more specific, e.g., *.AFT or TILEREF.AFT.

P 70 T 44 ITD (Interim Terrain Data) is not a good example for a library name, since ITD is itself a database.

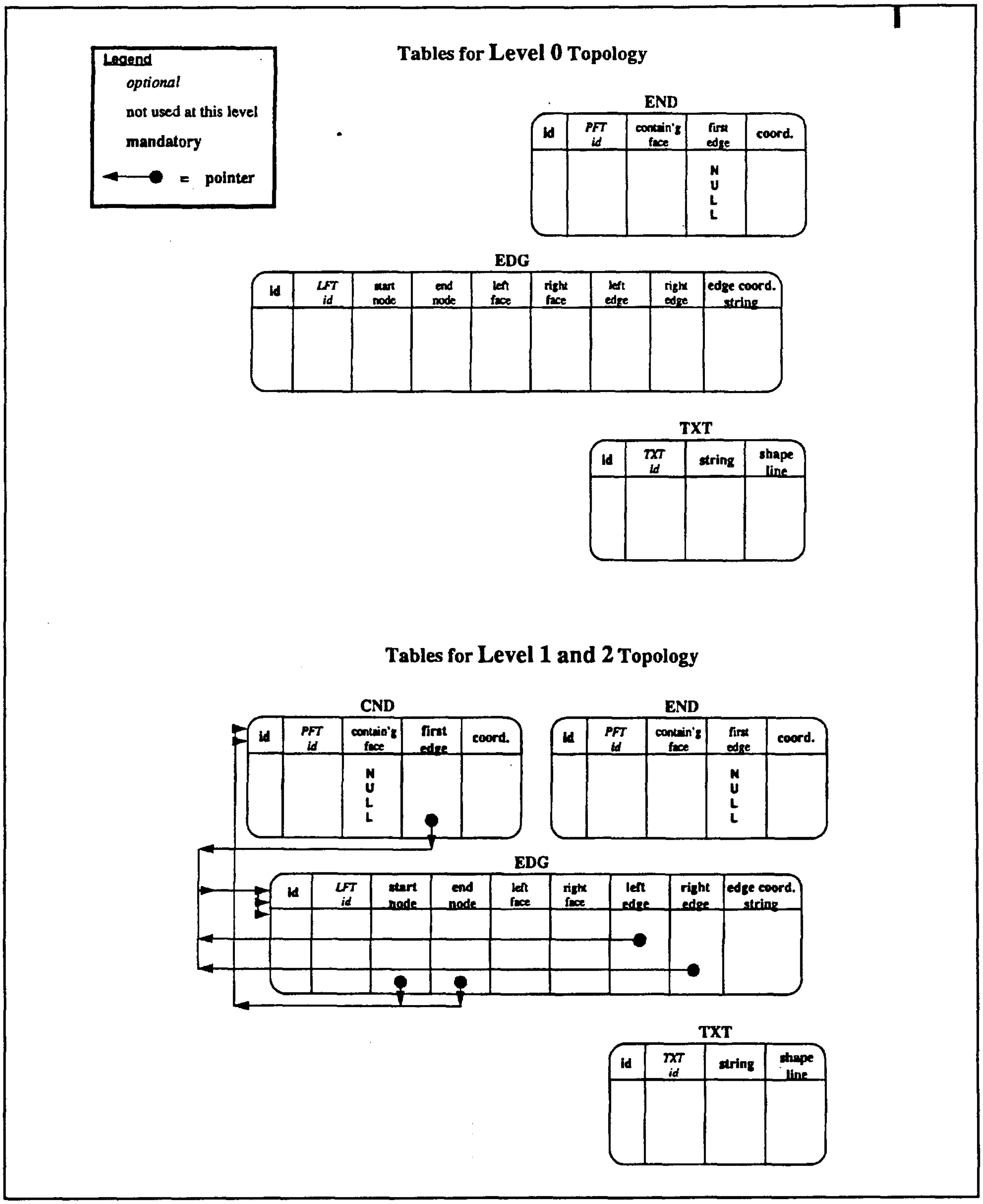

Figure 2. Suggested tables to show levels of topology.

Tables for Level 3 Topology

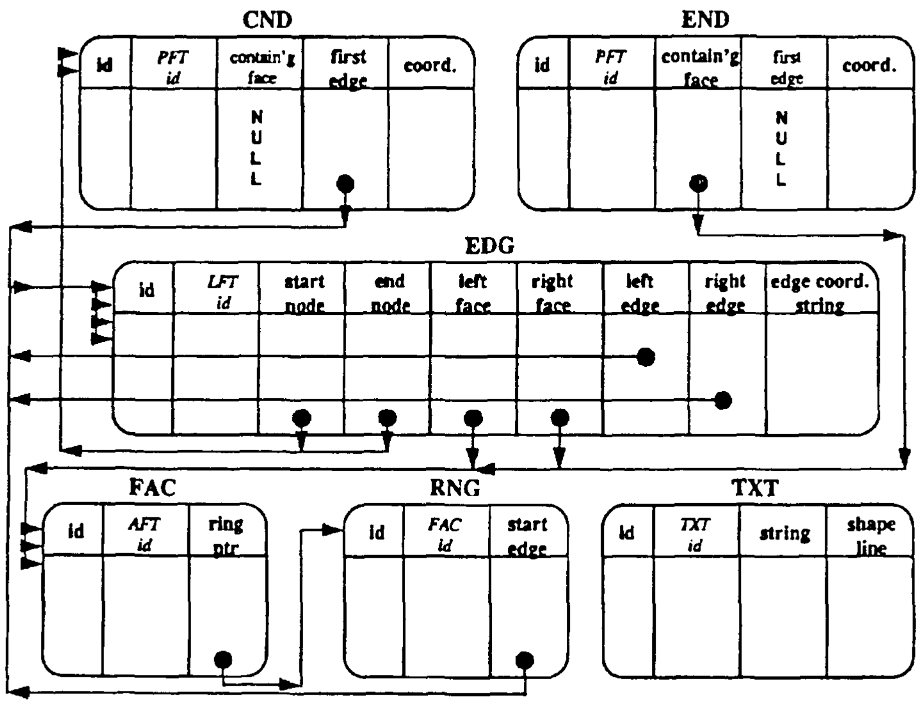

Figure 2. (continued)

P 83 L 11 — The exposition should be adjusted so that tables are distinguished from indexes. Suggested alternate text is as follows: "One feature join index can be defined for each of the five primitive types in a coverage. This index can be represented by a feature index table (*.fit). For example, an edge feature index table, edg.fit, can be defined for ..." As the paragraph now stands, the distinction between index and table is blurred.

P 166 — The figure in the beginning of Appendix H of [2] was an excellent way of portraying a sample VPF product and the structural levels of VPF. The figure (or a similar one, such as Figure 1 of this review) should be reinserted into [1].

3.0 Editorial comments

All editorial comments are included in the following list.

P iii L 1 — The CONTENTS section should be revised. The section titles should be indented, and the individual sections of the Appendices must be listed if the reader is to follow the text (e.g., Appendix C has a structure that needs an outline if the reader is to fully understand the types of joins and performance).

P 43 L 4 — Change "The reference coverage" to "This reference coverage."

P 46 L 23 — The fact that the names placement coverage is discussed should be indicated in this sentence.

P 48 L 6 — Change the last sentence to the following: "These optional feature pointers are recommended when tiles exist in the coverage."

P 62 L 12 — Change "on the feature table" to "in the feature table."

P 65 L 44 — Standardize on the way appendices are referenced: Appendix G or appendix G.

P 82 L 3 — USE CODE should be USE_CODE.

P 83 L 6 — Add the "A" at the beginning of the sentence: "A feature index may be ..."

P 83 L 11 — Change "features/primitive joins" to "feature/primitive joins."

P 122 F 36 — There is an extra word in the caption of this figure: *untiled.*

4.0 Recommendations

The "essential" comments noted above should be incorporated into the VPF specification. In addition, the introductory general comments should be considered. These changes will promote VPF user comprehension. Although of lessor importance, the "suggested" comments will nonetheless enhance the specification. Finally, the editorial changes listed in this review under Section 3.0 should be made.

DMAP's purpose in evaluating VPF was not to determine the format's acceptability. The current widespread use makes changing to a new format such as object oriented virtually impossible. However, the observations and modifications contained herein should enhance VPF without introducing burdensome changes.

5.0 Acknowledgments

This effort was sponsored by the Digital Mapping, Charting and Geodesy Analysis Program (DMAP), funded by the Oceanographer of the Navy under Program Element 0603704N. The DMAP Program sponsor is RADM J. Chesbrough (N096) and the NRL (Naval Research Laboratory) program managers are Mr. Ken Ferer and Mr. Harry Selsor.

Technical review of this report was provided by Mr. Mike Harris and Ms. Maria Kalcic, both of the NRL Mapping, Charting, and Geodesy Branch, and Ms. Mary Clawson of the NRL Marine Geosciences Division.

6.0 References

1. Defense Mapping Agency, "Draft Military Standard Vector Product Format," MIL-STD-2407, 30 September 1993.

2. Defense Mapping Agency, "Draft Military Standard Vector Product Format," MIL-STD-600006, 13 April 1992.

Appendix A. Acronyms.

AFT	Area Feature Table
DHT	Database Header Table
DIGEST	Digital Geographic Information Exchange Standard
DMAP	Digital Mapping, Charting, and Geodesy Analysis Program
DNC	Digital Nautical Chart
FACC	Feature and Attribute Coding Catalog
FACS	Feature and Attribute Coding Scheme
GEOREF	World Geographic Reference System
ITD	Interim Terrain Data
LHT	Library Header Table
NRL	Naval Research Laboratory
TILEREF	Tile Reference Coverage
VMap	Vector Smart Map
VPF	Vector Product Format
WVS	World Vector Shoreline

Appendix B. NRL DMAP Technical Review of VPF dated 4 October 1991.

NRL DMAP TECHNICAL REVIEW OF VPF

DMAP VPF Technical Review Team
Mr. Kevin Shaw
Mr. Jim Hammack
Dr. Cliff Burgess
Ms. Danette Coughlan

Background

The key technical issue associated with the VPF specification is the relational database model on which VPF is currently based. VPF is designed around this relational model as opposed to an object-oriented database model. DMAP's technical review of VPF focuses on contrasting these two database models. The network model and its special case, the hierarchic model, are not discussed in this review since they are not major database model contenders. The relational, network, and hierarchic database models are normally referred to as conventional or semantic database models. [1]

DMAP's technical review focuses on seven key areas to compare and contrast the relational and object-oriented models: database and programming language interface, data manipulation, supported data types, standards, database design methodology (defining the database structure), GIS usage requirements, and computational complexity. Conclusions and NRL DMAP recommendations are provided based on these key technical areas.

Database and Programming Language Interface

In many instances 30% more application code will be required to translate between the relational database model and the programming language, such as C++. Object-oriented databases do not require this additional overhead since languages such as Ada and C++ already conform to the object model. [2]

Data Manipulation

Structured Query Language (SQL) is used as the interface between the relational database model and the typical programming language. SQL and the relational model view the database in a logical sense as a table. This tabular organization is the key relational limitation with large real-world databases.

Using an object-oriented database model would allow the data to be manipulated in the same manner as data is manipulated by an object-oriented language like C++ or Ada.

Supported Data Types

The relational model only typically supports the following data types: integer, decimal, float, character, and string. There is no facility for users to define their own data types. However, with the object-oriented model, the full range of data types offered by object-oriented programming languages are supported, including user defined (sometimes called enumerated). [3]

Standards

In the last couple of years, the primary standards body defining the International Standards Organization data structures for design data, the Product Data Exchange Specification (PDES) committee, has moved firmly in the direction of object-oriented specifications. [4]

It is clear that the DoD standard programming language, Ada, is an object-oriented language. Also, C++ is currently being heavily used and it is also based on the object model.

Database Design Methodology (Defining the Database Structure)

Relational database design is built on fitting database content into a number of rows and columns (called records and fields). This is a severe limitation when the amount of attribution is large. For example, DMAP experienced structure and query limitations with dBASE IV using the data collected from the requirements analysis study. The requirements analysis database is not nearly as complex as a fully populated DNC database will be.

Object-oriented database design would be an integral part of the application software development cycle. Therefore, the database structure would not be limited to fitting the real-world data and attribution into a tabular (record and field) structure. The database would be viewed as another section of memory to be accessed by the application software.

The structure of the database is critical since it determines how and if various queries on the data will be possible.

GIS Usage Requirements

On 19-22 July 1989, the National Center for Geographical Information and Analysis (NCGIA) held the specialist meeting of the research initiative on Very Large Spatial Databases (VLSDB) in Santa Barbara, CA. At this workshop, 42 participants from the U.S. and Europe discussed issues related to GIS usage of VLSDBs. One of the meeting's conclusions was that the object-oriented approach was a helpful tool in improving the modeling of geo-objects in VLSPDs. (Note: NCGIA is an NSF-funded institution not linked to any commercial database vendor.) [5]

On 12-13 October 1990, the NCGIA met again to discuss GIS issues again with emphasis on temporal relations. It was noted that the object-oriented model is the most useful mechanism available for modeling space and time. [6]

Computational Complexity

An object-oriented model based database has been shown to reduce the number of components that are operated on and thereby reducing the computational complexity. [7]

An accurate performance comparison between these two database models requires the same database to be implemented in conformance to each model. Also, the manner in which the relational and object-oriented model are implemented will affect the various access and storage times.

Conclusions

The relational and object-oriented database design approaches are very different. In the relational case, real-world objects are forced into a tabular construct without regard to the "natural" organization of the data. This forced fit then propagates forward when a programming language is required to interface with this tabular database structure. The programming environment is complicated since there is a stand-alone database model and a programming language based on a different model (C++ and Ada are based on the object-oriented model). However, in the object-oriented case, real-world data is grouped into real-world objects. This presents a consistent programming environment where the database and the programming language are utilizing the same model. In actuality, the object-oriented programming environment does not have to separate the programming language and the database management system. These two components can be viewed as simply utilizing different partitions of memory (the program operating in its memory space pushing and popping data from the database memory space). This consistent environment then allows the user to optimally query the database information based on real-world scenarios with fewer constraints.

The recent DMAP requirements analysis study showed at least 28 Navy programs are using Ada and many more C++ (both object-oriented languages). The optimal programming environment for Navy programs utilizing Ada or C++ should include a database based on the object model as the programming language.

The commercial market's leading relational database management system, dBASE IV, has recently announced that dBASE IV will be moving from the relational model to an object-oriented approach. When this is combined with the heavy usage of C++ and Ada, it is clear that the object-oriented model is the preferred approach.

The relational database model can be extended to provide some of the capabilities found in the object-oriented model. However, there is still an inherent mismatch between the extended relational model and the truly object-oriented model. [2]

Recommendations

The object-oriented model is recommended for large real-world spatial vector databases that are heavily attributed, e.g., DNC. If all DMA vector products will eventually be stored in a single standard format (like VPF), then only an object-oriented model offers the flexibility to support the levels of attribution required for the more complicated worldwide databases. This is critically important since the recent DMAP requirements study showed that 58% of Navy programs require worldwide coverage. Currently, DCW is the primary database stored in VPF. The problems associated with the relational database model have not been overwhelming with DCW since this database is not heavily attributed and is at the 1:1M scale.

In the near term, it is anticipated that a full implementation of the DNC in VPF will have severe difficulties with the large amounts of attribution and the overall size of the spatial database. The end users of the DNC will not have the flexibility to retrieve real-world objects in an efficient manner if the DNC as well as other large spatial databases are based on the relational model, i.e., stored in the current version of VPF.

The programming language and the database management system used should not be based on separate models. For example, programming in C++ or Ada (object-oriented model) and using a dMC&G database stored in VPF (relational model) creates a model mismatch. The programming environment should consistently use the same model. It is clear that programming languages have moved to the object-oriented model and very likely that the successful database management systems in the future will be object-oriented.

Additional Navy and DMA study is required on storing large heavily attributed spatial databases in a format based on the relational database model, like VPF. This study should be completed before a decision is made to store all DMA vector databases in the VPF. At this time, the object-oriented model approach seems to offer more potential for a single robust vector database standard when compared to VPF (a relational model implementation).

If it is not possible to move to a truly object-oriented model, DMAP recommends extending the relational model to include as many capabilities of the object-oriented model as possible. This would allow much more flexibility compared to a purely relational model, as seen in the current VPF implementation.

References

[1] Alfonso F. Cardenas, "Data Base Management Systems," Second Edition, Allyn and Bacon, Inc., 1985.

[2] J.V. Joseph, et al., "Object-Oriented Databases: Design and Implementation," Proceedings of the IEEE, Vol. 79 No. 1, January 1991, pp. 42-64.

[3] Mary E. S. Loomis, "ODBMS vs. Relational," Journal of Object-Oriented Programming, July-August, 1990, pp. 79-82.

[4] Christopher M. Stone and David Hentchel, "Database Wars Revisited," BYTE, October 1990, pp. 233-242.

[5] Terence R. Smith and Andrew Frank, "Report on Workshop on Very Large Spatial Databases," Journal of Visual Languages and Computing, 1990, No 1, pp. 291-309.

[6] Renato Barrera, et al., "Temporal Relations in Geographic Information Systems: A Workshop at the University of Maine," SIGMOD RECORD, Vol. 20, No 3, September 1991, pp. 85-91.

[7] L. Mohan and R. L. Kashyap, "An Object-Oriented Knowledge Representation for Spatial Information," IEEE Transactions on Software Engineering, May 1988, pp. 675-681.