TOMPA, FRANK WM.

The application of current database technology to videotex.

THE APPLICATION OF CURRENT DATABASE TECHNOLOGY TO VIDEOTEX

Frank Wm. Tompa

Jan Gecsei

Gregor V. Bochmann

June 1981

This report is prepared under contract 20SU.36100-0-9506 between the University of Montreal and the Department of Communications, Canada.

The opinions expressed in this report are those of the authors. They do not necessarily imply any position of the Department of Communications.

1. Introduction

A central part of a videotex system is its store of information in the form of a videotex database. Many of the benefits sought from traditional business-oriented databases are also desirable goals for videotex; among these are reduced redundancy, reduced inconsistency, increased sharing of data, increased potential for standardization, enforced security, representation independence, increased protection from user and system errors, and improved performance. It is therefore natural to investigate the current state of videotex technology in order to learn how to provide a good videotex system.

We shall start by examining the similarities and differences between traditional corporate databases and videotex databases, concentrating on those properties that separate videotex from traditional business applications. Next we will investigate the facilities provided by traditional database management systems and describe each facility's applicability to a videotex system. As a third step, we will summarize the state-of-the-art architecture for a database management system (in particular, we will describe the interfaces proposed by the Ansi/X3/Sparc Study Group on Database Management Systems [Ansi 75]) and report on its applicability to videotex systems. After that, we

will investigate how videotex systems can be used to access databases that have been created and are maintained primarily for purposes other than videotex.

Before starting, a question that arises is "what makes videotex database management unique?"

(1) Future videotex data will be supplied by a large, diverse community of (competitive and cooperating) information providers. There is likely not to be a "corporate enterprise view" of the data as can be expected in most business's applications. Furthermore there is likely to be a demand for a variety of presentation styles to suit the individual editorial tastes of the information providers.

(2) The database will be accessed by an extremely large number of simultaneously active users. Most database applications now speak of hundreds or at most one thousand users, whereas videotex systems may be used simultaneously by 10% of the households in its service locality (e.g., 50000 homes in metropolitan Toronto).

(3) Unlike traditional corporate databases, the vastness of the data will result from its diversity rather than from a high multiplicity of a few well-defined types of data. Whereas videotex may include data about cars, sports scores, games, finance, and arbitrarily many other topics, it is unlikely to contain individual data on more than a few hundred entities of each type. This is

in contrast to corporate databases having millions of records all of the same format and containing similar data.

(4) The data format will remain to be constrained to fit closely to the display medium, which will allow multi-colour graphics and text to be intermixed. In particular, the data will be partitioned into screen-sized pages, although multi-screen and multi-media sequences may become common. Such data format reliance on external media is not as common in current corporate databases, where typical systems goals are to provide facilities for keeping the data's structure independent of display.

(5) Interactive videotex, by definition, relies heavily on two-way communication. This is witnessed not only in the spontaneity of page requests, which closely parallels interactive queries against traditional databases, but also in the availability of transaction invocation via response pages and of other action pages for executing pre-packaged or user-defined programs. These forms of interaction parallel the facilities of transaction systems and time-sharing systems rather than traditional database systems.

These features of videotex will be seen to have a significant impact on the requirements for a good videotex database system.

2. Videotex pages

The primary data unit in today's videotex systems is the page. Each index or document is formatted into parts that fit within one screenful of characters (e.g., 20 lines of 40 characters each, possibly with graphics superimposed on a logical grid having these same dimensions). The Telidon system augments this capability to include some animation by allowing further data to overwrite some of the screen's display, but the logical unit is still primarily a single screenful of information.

In most current videotex systems, any page of data that appears on a user's screen is, in fact, stored as such in the database. That is, even if two or more pages have a display with much of the data in common, that data is stored separately for each page rather than being shared. As described in previous publications [Ball 80b, Tompa 81] and presumably implemented in some external computers for Bildschirmtext [Otto 81a], an alternative method is to store blank forms containing common data and to fill the forms with a page's particular data values only when that data is actually requested by a user. Thus pages need not necessarily be stored individually, but rather they may be assembled on demand. For example, an airline's flight information may be recorded in conventional records with fields for flight number, source, destination, departure time,

etc., and the data for a particular flight may be inserted into a blank "flight information page" only when a user requests that particular logical flight page. As a second example, a computer-assisted instructional page may be dynamically created by the system from a collection of stored fragments in order to re-inforce a particular aspect of a course needed by the student currently using the system.

A second aspect of dynamic page creation lies in the ability of videotex users to update the database directly. For example, the data in a "want ads" page (or set of pages) should be updatable on-line, as new items become available and old ones are sold. This form of on-line update is already demanded by some information providers, and it will likely be of even more use if available to all users (under very restrictive circumstances, of course). Again, the form-filling mode of operation may be useful here. In Bildschirmtext, for example, a blank form is displayed by the system, the user inserts the data values into pre-defined areas, and the (external computer's) database system may extract the fields to be stored independently as above or it may choose to store the pages as filled [Otto 81a].

In the next section, the facilities available in traditional database systems will be examined and evaluted

with respect to their utility for managing these three modes: standard stored pages, dynamically assembled pages, and dynamically updated pages. Such a survey may help to direct the design of future videotex systems which may offer pages using any combination of these modes.

3. Database management system facilities

3.1. Traditional corporate database management facilities

Corporate database management systems have been developed over several years to meet the needs of most businesses. At least superficially, it would seem that videotex systems should provide similar services. In this section, traditional database facilities are categorized into three areas: data modelling, system functionality, and implementation methods to support the first two areas.

3.1.1. Traditional data modelling

There have been many data models proposed by researchers from the database community [Kerschberg 76], most common of which are the hierarchical, network, and relational models [Sibley 76]. Although there are many differences among the models, they all presuppose an interpretation of reality in which there are classes of individual elements (e.g., entities), relevant attributes or properties of the elements, and relationships among the ele-

ments. Furthermore, the data space is partitioned into types where the number of types of elements, attributes, and relationships is small relative to the number of data instances (e.g., actual data objects) [McLeod 81].

Each data model requires that a data base designer define a schema, which specifies the time-invariant structure imposed on the types. Typically an instance of one type can be related to many instances of another type, but the relationships must all be pre-defined as part of the type structure. As a further restriction, the traditional data models cannot represent a data relationship between one instance of a type and arbitrarily many other instances of the same type, or they represent such relationships poorly. As a symptom of this restriction, a query that is typically difficult to accomodate using traditional data models is "find all employees who earn more than their managers" (where managers are themselves employees too). The result is that a traditional data model may have only limited applicability to videotex (see Section 3.2.1).

3.1.2. Traditional database system functions

Corporate database users fall into four categories: data administrators, system administrators, applications administrators, and end users. In order to serve these users, traditional database systems provide the functional components summarized in this section and in Table 1.

data definition language

data dictionary facility

data manipulation language

report generators

query language

data storage definition language

performance measurement and evaluation tools

other operating subsystems
(including those for integrity, security, input/output, memory management, processor allocation, terminal management, and accounting)

Table 1: Traditional functional components and support aspects

A data administrator deals with the database at an abstract level, defining types, integrity constraints, security restrictions, legitimate usages for the data, etc. The primary tool for this purpose is a data definition language. Constructs in the language allow the specification of element types as well as their attributes and interrelationships. Traditional data definition languages also have facilities for specifying security levels for data objects and users and at least a primitive version of integrity constraints in the form of data validation rules. In many database systems, the definitions are compiled and catalogued in a data dictionary that is available to all authorized database users to enquire about the nature of the data and to gain further information regarding the intended semantics and pragmatics of the database.

The role of a system administrator, on the other hand, is to maintain the database management system itself as well as to tune the internal data structures in order to minimize some measure of cost (e.g., space, processor time, channel utilization, security). This role corresponds to that of a system operator in a videotex system. The language used by a system administrator, sometimes known as a data storage definition language, is closely related to (and sometimes a part of) the data definition language. Typically this language is used to specify precisely how the data is to be represented, for example, what block sizes to

use, how many input/output buffers to allocate, and how to link records together. Some of the options are discussed further in the next section.

Each application administrator has the responsibility of defining an appropriate view of the database to present to a subset of the end users, typically also providing some software for those users. This is the role of the information provider in videotex. Application administrators are also users of the data definition language, through which they are able to specify the subset of the database that will be available to their associated end users. As a further specification of end users' views, the data definition language provides means of specifying display formats; renaming entities, attributes, and relationships; introducing "virtual data" whose values are calculated dynamically from stored data; and further limiting data access through security and integrity constraints.

Two other facilities are also used extensively by application administrators: a data manipulation language and report generators. The first of these contains constructs to load the database, access specific items, manipulate data values, insert and delete entities, and update their properties and relationships. The report generators are used to specify complex report formats and to

indicate how the database is to be processed to determine the values to be included in the reports. Both these facilities are typically invoked through the use of a computer programming language, and they therefore require their users to be data processing specialists.

Finally the end users are the ones responsible for populating the database, modifying data values, and retrieving the data required to solve particular problems. They typically interact with the database through an interactive query language [Ball 81, Lochovsky 81]. This facility is usually designed for computer non-specialists and is therefore relatively easy to learn and to use. Typically the query language allows simple, spontaneous interrogation of the data and the dictionary, simple data editing, and invocation of prepackaged data manipulation and report generation routines (as provided by the applications administrators as above).

3.1.3 System support facilities

The most apparent support facility is that which allows a selection of data and file structures for representing the records and their links, typically a major part of the data storage definition language. Several structures are commonly used [Wiederhold 77]:

A pile is an unordered collection of records, used

primarily for sequential processing as for audit files.

A sequential file is ordered by some specific data field, called the key, and typically used for batched transaction processing.

An indexed file is a collection of records together with a (multilevel) index that provides rapid access to the individual record containing a specified key value. One special case is the indexed-sequential file in which the underlying collection is kept sequentially by key value; in this situation, the index is often represented by a B-tree [Comer 79].

A direct file provides access to records by record number. This can be viewed as another special case of an indexed file, where the indexing operation is computational (i.e., the key values are themselves record numbers or they can be transformed to record numbers by a hash) rather than based on a search strategy.

An inverted file provides access to records based on values in data fields other than the key. Indexes are provided for arbitrarily many designated secondary keys, which do not necessarily identify records uniquely, but rather partition the records into subsets having common secondary key values. Three typical representations keep the underlying file as a direct

file, as superimposed multilists, and as superimposed multirings, respectively.

In addition, data compression can be used to save space and record transfer time, either as part of or auxiliary to any of these structures [Gottlieb 75].

A second class of support tools are those required for performance measurement and evaluation. These include tools needed to collect performance data, as well as to model and analyze the database system (e.g., by simulation or analytically) [Infotech 77].

Finally a database management system needs many of the utilities that can be found in a standard operating system. Of particular interest are integrity, security, and input/output subsystems, but also of importance are primary and secondary memory management, processor allocation, terminal management, and accounting subsystems.

All the aspects listed in this section are required by videotex system operators in a form very similar to that used in traditional data base systems. Therefore they will not be considered further in this study.

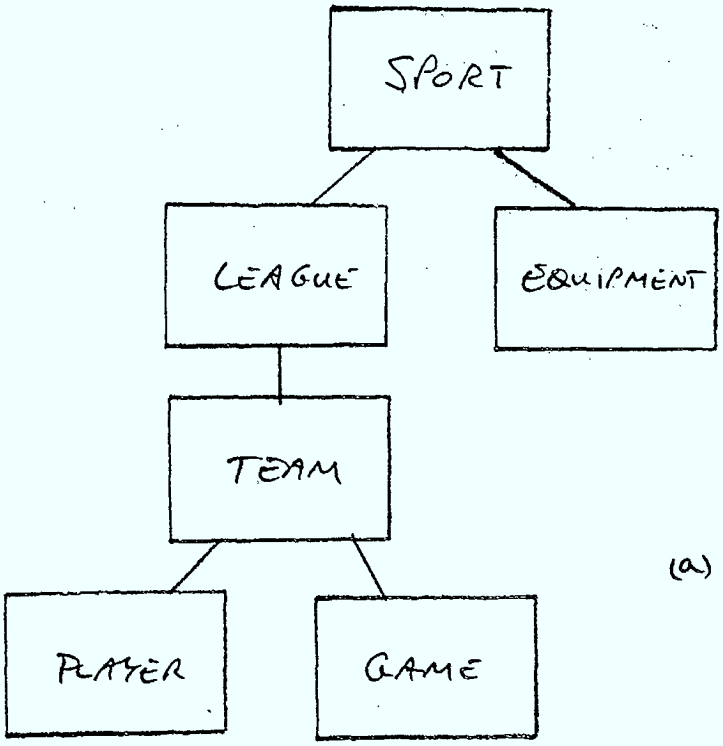
3.2. Applicability to standard stored pages

Current videotex databases are all built upon a tree of pages over which an arbitrary directed graph can be

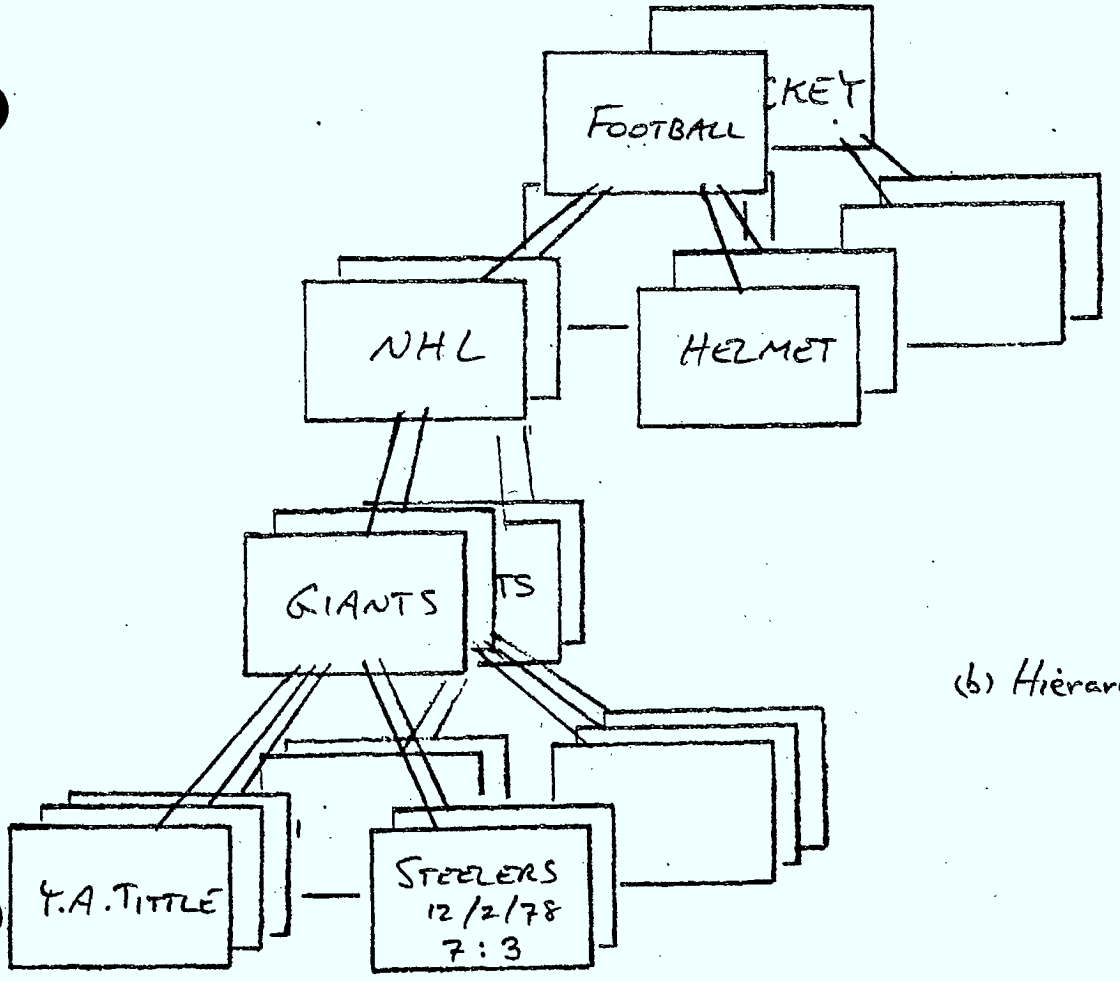
imposed by the information providers [Tompa 81]. Users search the database by means of requesting individual pages, typically by their (absolute) page identifiers, their (relative) cross-link labels, or their (relative) position in the tree.

It is tempting to view a tree of pages in terms of a standard database model, particularly the hierarchical one. There are two significant problems, however: the identification of types in videotex pages is artificial, and the operations on pages are unlike typical operations on conventional corporate databases.

If a tree of pages is to be viewed as a hierarchical database, the type structure must form a hierarchy; that is, after the pages are partitioned by type, the graph that represents the relationships between instances of the types must be a collection of trees (see Figure 1). These trees are very unlike the videotex tree of pages in that a hierarchical database tree represents a larger collection of instances, where the descendants of an instance node form arbitrarily long sequences of instances of each descendant type declared in the type structure. This has not been the mode of data presentation common in videotex to date, where page format and content varies considerably among descendants of a page and there is a very small limit on the number of descendants permitted.



(a) Hierarchical schema



(b) Hierarchical instance

Figure 1: Hierarchical Database

The second problem with applying traditional models to videotex page structures lies in the operations that are typically employed to search and manipulate the data. A typical query in the conventional database illustrated in Figure 1 might be a formulation of "for each team in hockey's NHL league, list the dates and teams played in 1979 together with the score of each game and the total won/lost record," whereas in videotex systems it might be "show me the data on the last Toronto/Montreal hockey game." That is, whereas conventional data processing deals with sets of records, videotex processing deals with individual pages one at a time and relies on a different user strategy for finding information.

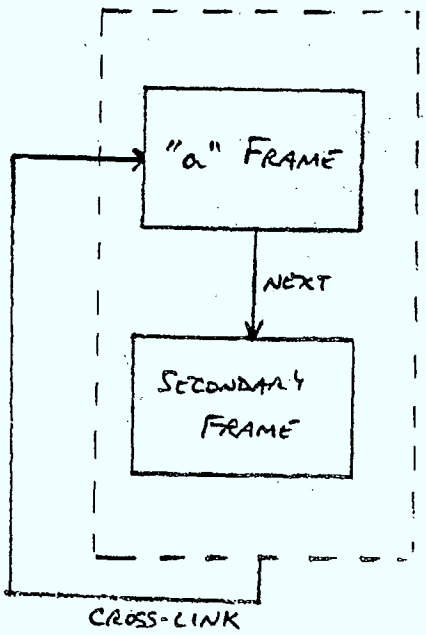
A conclusion from the discussion so far might be that traditional database management and videotex have little in common. This may very well be true for the models, and therefore traditional database management systems cannot reasonably be used to support the stored pages of today's videotex databases. However, the functionality indicated in Table 1 from Section 3.1.3 may still be relevant to the design of videotex page management systems, as we will show in the rest of this section. Subsequent sections will show that such facilities, as well as the data models, are even more applicable under dynamically assembled and dynamically updated page systems.

Data definition language:

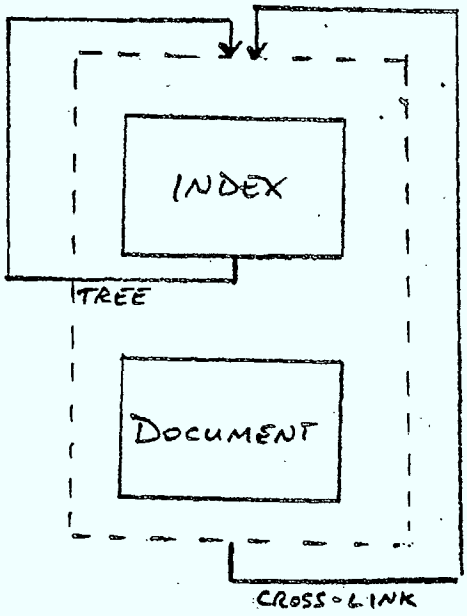
Because the data in videotex is expected to be so wide-ranging and its structure is to contain far less repetitivity than is typical in traditional corporate databases, the utility of a data definition language is unclear if pages are always stored intact. Certainly, some tool is necessary for an information provider to describe the data layout and structure, but it may be that there is no need to describe a schema independently of the instances.

Rather than dismissing the utility of a data definition language altogether, we illustrate three levels of development for page types to show the potential for a data definition language in some videotex systems.

First we examine a minor type distinction seen in current videotex systems. In Prestel there are two page types, "a" page frames and secondary ("b" through "z") page frames, the first of which can be accessed by cross-links from any page frame and the latter of which can only be accessed sequentially from a corresponding "a" frame. In Telidon there are again two page types, index and document pages, either of which can be accessed by cross-links from any page or by embedded tree links from index pages (which induce a sequential ordering). Corresponding type diagrams are shown in Figure 2, where dashed boxes indicate union



(a) PRESTEL type structure



(b) TELIDON type structure

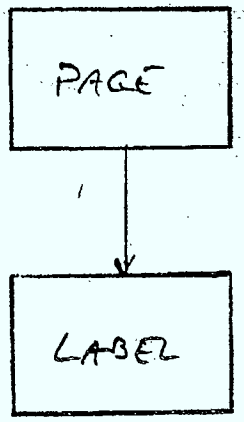
Figure 2: TYPE STRUCTURES IN TODAY'S VIDEOTEX

types.* (This illustrates the point made in the introduction to this section: the inter-page structure in videotex is unlike the inter-record structure in traditional corporate databases.) In current systems there is no need for a data definition language: the type structure is invariant.

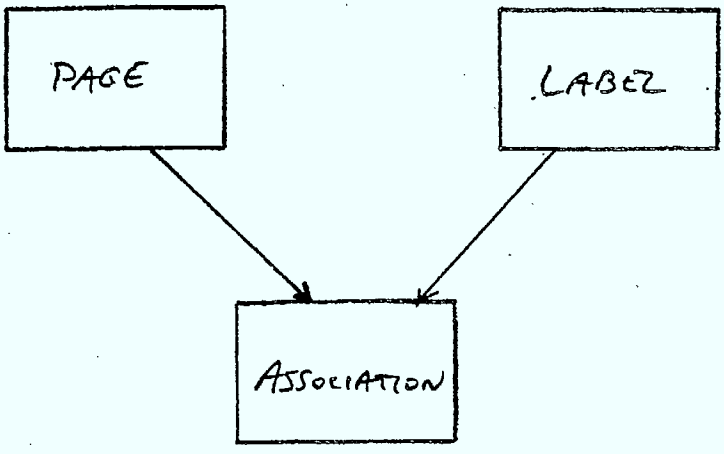
The use of keywords to identify pages may also be interpreted as requiring a separate data type. For example, a mnemonic labelling scheme for pages, where each label identifies at most one page and each page has one or more labels, could be interpreted as using the schema in Figure 3a, whereas many-to-many keyword-to-page associations would be as in Figure 3b. A data definition language could be used to identify the form(s) of keywording available in parts of the database, but, again, it would not be very useful if an invariant strategy for keywords is to be enforced.

As an alternative approach, it may be useful to distinguish page types based on their use within "page clusters" that commonly appear as a result of editorial policy. For example, it may be useful to distinguish a hub page type from a rim page type in a "wheel structure" [Taylor 79] in order to allow the system to enforce integrity constraints during page updates. Such a distinction could also be valuable to end users to aid them in understanding the configuration of a cluster of pages (in some

* In data structure diagrams, an arrow indicates a one-to-many relationship [Bachman 69].



(a) Mnemonic labelling



(b) Keyword-to-page association

Figure 3: Type Structures in Keyword-Videotex Systems

neighbourhood) and to navigate the cluster meaningfully. On the other hand, the variety of structures in videotex and the limited repetition of each structure make the requirement of defining a schema cumbersome. Thus a data definition language may only be of limited use: we suggest that information providers be allowed to define page types where desired, but they may more often prefer to use untyped ("blank") pages.

Data dictionary facility:

The purpose of a data dictionary is to document the data that is kept in a database -- it contains data about the data. In traditional corporate databases, the dictionary typically contains information about the entities, attributes, relationships, processes, and people that make up the enterprise. This information is accessible by both database administrators and end users in order that meaningful queries and updates can be formulated and responses can be interpreted within the framework intended by the enterprise administrator.

A similar role is to be played by vidotex's metaservice, a directory of information in a videotex database [Le Moign 80, Ball 80b]. In the Teletel/Star system, service selection is performed before normal page selection begins, but there is no reason this separation must exist.

Just as the traditional data dictionary is under the strict control of some central authority (for example, that of the data administrator), the service directory must be centrally managed to ensure its integrity. As well, in parallel with a traditional data dictionary, the service directory must be maintained and made available by all appropriate users. In particular, the system operator and each information provider must cooperate to keep the service directory accurate, complete, and up-to-date, so that the contents form a true metaservice for all users.

The metaservice should contain entries corresponding to each information provider and each subject area. In addition this service directory must contain the data needed to access the pages corresponding to each specific information provider and to each keyword or subject.

Further discussion of the metaservice is contained in Section 4.2.

Data manipulation language:

The primary tool for populating the database, manipulating the data, and formulating complex queries about the data and the database itself is the data manipulation language. As mentioned in Section 3.1.2, this language is typically used by applications administrators rather than by

end users, who rely on the query language.

Such a facility in videotex is required by the information providers, who are not computer professionals as are the typical users of traditional data manipulation languages. Furthermore the standard modes of processing corporate records are not those of processing videotex pages. Therefore a videotex data manipulation language must be specially designed to suit its intent.

The first aspect of this language is for populating the database. Certainly pages can be created individually as they are typically created now. However there will definitely be a need to extract data from established databases or other external sources and to build pages under computer control. For example, the automatic incorporation of the Associated Press wireservice into the Department of Communications' database is a special case of this facility. What is needed to make such an interface easier to establish and maintain is a language that permits the definition of the source data, the definition of the desired page-oriented version, and the algorithms to effect the loading.

A second aspect is the editing of the interpage structure. There must be commands to allocate "blank" pages for storing new data; creating, altering, and removing interpage links; displaying the current interpage structure;

and releasing pages (for example, for future reallocation). Finally, if such editing is to take place on-line, there must be commands to signal the start and end of a multi-page transaction, in order that concurrency control can be maintained.

A third aspect of a data manipulation language is the facility to retrieve information satisfying relatively complex queries. The most obvious query type is for data stored in the pages; page-by-page retrieval (as is currently used) may be satisfactory, but certainly more sophisticated strategies (perhaps based on keywords) should be considered. An additional form of query concerns data about the database itself, for example, how much demand there is for certain pages, how much space is used by certain pages, etc. Possible videotex data manipulation language interfaces (as well as the query language facilities) are discussed elsewhere [Ball 81, Lochovsky 81].

Report generators:

Because report generators are mostly used to organize and format highly repetitive data, there is little need for such a facility in a stored page videotex system.

Query language:

The design of a query language for videotex is central to its acceptability to end users. This facility is

therefore discussed at length in an accompanying report [Ball 81].

3.3. Applicability to dynamically assembled pages

Dynamic assembly of pages is a useful feature when there is sufficient sharing of information among pages such that either the storage saved is significant or the consistent updating of the shared data is greatly simplified (because of less redundancy). In this section, it will be assumed that a page is typically composed of several components stored separately.

The primary effect on the videotex database system is to increase the utility of types. For example, a large number of pages may be defined as aggregations of instances of the following types: page number, information provider name, price, text, logo, master index page, previous page, and next page (Figure 4). Rather than storing the layout, colours, and constant data for each page separately, this information can be factored out into a form type that is dynamically filled from data stored in a more compact way.

An extremely beneficial use of dynamically assembled pages results from the provision of a record-oriented database facility integrated with a videotex database system, as in Bildschirmtext. As mentioned in Section 2, a blank form can be defined as a videotex page whose

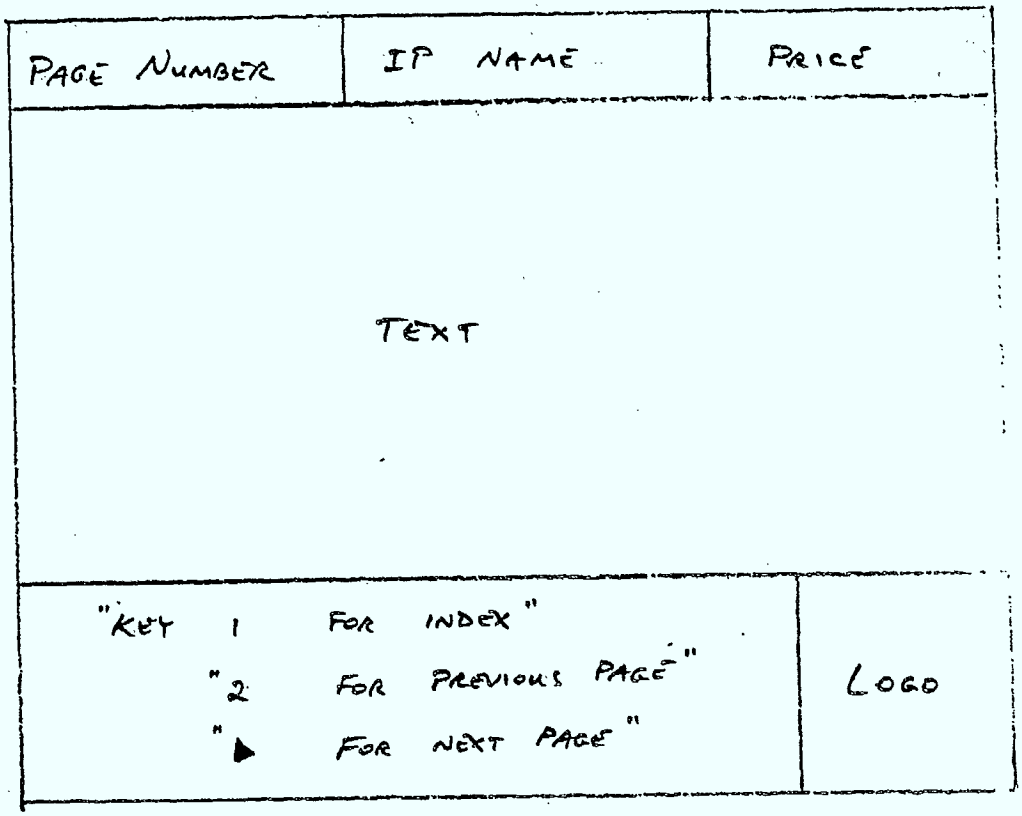


Figure 4: SAMPLE FORM TYPE

contents is determined only when data is requested by a user, that data being extracted from the record-oriented database. The role of a database schema is obvious for the records, and it extends naturally into the definition of form types for videotex pages. This sort of facility is similar to the forms languages available in some traditional database systems, for example, QBE [Zloof 75] and Oracle [RSI 79].

Many aspects of managing a videotex database with dynamic page assembly are similar to those required in traditional database systems. In particular, the need for data definition, data manipulation, and data storage definition languages is increased over the standard stored page systems. It still must be remembered, however, that the multiplicity of the data instances of one type will remain to be much smaller than in conventional databases and the usual mode of processing will be navigational in nature; these differences must be kept in mind during the language designs. As a particular example, severe constraints on retrieval time may be imposed by the concern for economical access to videotex; thus, processor-consuming operations, such as the relational "join", may be unsuitable for videotex data retrieval.

3.4. Applicability to dynamically updated pages

As for all videotex systems, the major, or only,

language available to end users is the so-called query language; this, rather than the data manipulation language, is therefore the major design problem for a system that will allow users to update pages interactively. Because most end users will be non-specialists with respect to computing, the language must be simple, yet complete and safe to use. Form-filling languages, such as used in Bildschirmtext, are probably the most natural for naive users; and with each field in the form there can be associated data validation rules to ensure data integrity. Regardless of the update facility chosen, it must not be possible to enter inconsistent or invalid data through the end user facility even when the user takes no explicit precautions.

Consider when the desired update is specified by filling in a form provided by the information provider (cf., "data gathering" in Bildschirmtext). The form itself may have several parts, each requiring data to be entered or choices to be made. Before the database is actually updated, however, the changes specified must be validated by a program (or perhaps a set of rules presented as a decision table) supplied by the information provider. Invalid entries must then be flagged to the user, who refills the fields and retransmits the form. If form-filling is provided by a special-purpose action page or by any other means of interleaving code execution with data display (e.g., as in Teletel/Star [Henriot 79]), the user's entries

can be validated incrementally rather than only when the page is completed. The impact of incremental checking vs. complete page validation on performance (in terms of the effect of field-at-a-time rather than complete page transmission from terminal to host and the number of data fields retransmitted during error correction) should be studied before one of these alternatives is chosen.

One problem in multiple, simultaneous updating is the coordination of several users. In order to avoid the necessity of intricate update protocols, the unit of update should be one page (or one part of a page if dynamic assembly is used). In order to change a page's contents (e.g., to add further "want ads") the user first requests the page for update, indicates the change(s) to be made, and then either directs the actual update to be carried out or to be aborted. Upon the initial page request, the page is locked to all other readers and writers, and it is not made available until the command to complete or abort is issued (or the user's session terminates). Meanwhile the user should be barred from accessing other pages, either for reading or updating; such a restriction on the user's operations insure that a page is locked for only a brief time and there can be no deadlocks in circularly locking pages.

The burden left on the information provider is to organize the data such that it is reasonable to update the

pages independently. In particular, because only one page is locked and modified at a time, there is no way to ensure that a multi-page transaction is performed as an indivisible operation. It should be noted, however, that this restriction can be somewhat lifted if a single form can be used to indicate updates that affect several pages all of which are locked by the information provider; such a relaxation, however, does incur a computational cost as well as introducing the possibility of deadlock if not well-monitored. (For more complex locking strategies, not likely required for videotex systems, see [Gray 79].)

4. Database system architecture

Another area of research and experience in database systems that may be applicable to videotex is the design of the overall database system architecture. It has been generally accepted that users should be able to deal with a data model or view which is distinct from the storage representation adopted by a database system [Mealy 67, Senko 73, Tompa 77]. This so-called "two-level architecture", in fact, was embodied in the Codasyl recommendations for database design in the form of a schema for data representation and subschemas for users' views [Codasyl 71].

In 1972, the American National Standards Institute subgroup known as the Standards Planning and Requirements

Committee within the Committee on Computers and Information Processing (Ansi/X3/Sparc) began to consider areas for standardization of database systems [Ansi 75]. The result of their deliberations was a proposal for a three-level architecture including forty-two identified interfaces, only some of which were seen to be candidates for standardization.

In this section we will consider the recommendations of the Ansi/Sparc report and its application to videotex database systems.

4.1 Overview of a three-level architecture

A clear introduction to the three-level architecture, administrators' roles, and process interfaces was presented at the Second SHARE Working Conference on Data Base Management Systems held in Montreal in April 1976 [Yormack 77]. We present here a summary of the most important aspects.

A database should be perceived to exist in three principal, typically distinct, forms as described by a conceptual schema, an internal schema, and a set of external schemas (Figure 5). These descriptions are managed by one or more people serving in the roles of an enterprise (or data) administrator, a data base administrator, and applications administrators, respectively.

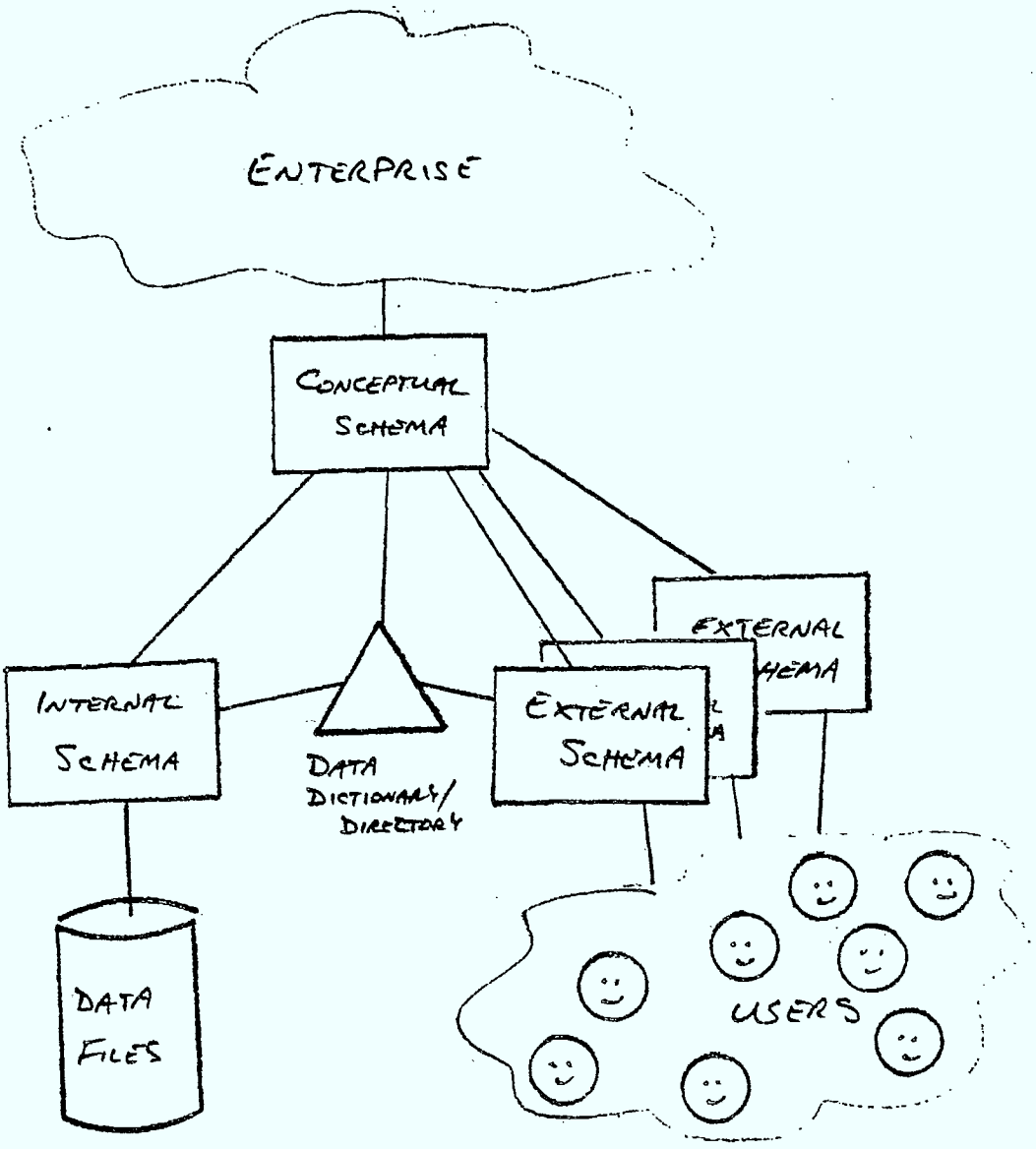


Figure 5: Three-level Architecture for Databases

The conceptual schema is best described by Yormack:

> "The conceptual schema embodies the 'real-world' view of the enterprise being modelled in the database.... It provides an unconstrained, long-term view of the enterprise. In addition, the conceptual schema provides the basis for integrity and security declarations imposed by the enterprise onto the various data base users, as well as providing a data description basis for restructuring. The conceptual schema also acts as the common denominator between optimized storage descriptions and multiple-user views." [Yormack 77, page 3]

The major point is that the conceptual schema provides a central focus for the purpose of harmonizing the needs of all users and the actual data representation. This harmony must be maintained at all times in spite of any reorganization of data representations, alterations of physical equipment, revision of user requirements or needs, users' contention for data access or modification, and evolution of the enterprise's procedures or goals.

The internal schema is a description of the data as it is represented in storage. It is this schema that records the current data placement strategy, access paths in terms of data linking organization, data redundancy and data compression aspects, and storage media characteristics for the database. In fact, it is only in terms of the internal schema that the database as a whole is stored, the other schemas serving as templates for presenting parts of the data as they are required by users and administrators.

Finally, it should be noted that the internal schema need not be a monolithic description of the data, but may in turn be composed of several levels of description from relatively representation-independent access path descriptions to minute, machine-dependent bit assignments [Tompa 77].

Each external schema describes a view of the data as seen by a set of users sharing common application programs. Typically, such a view provides a window's image of a subset of the data base. Furthermore, the image may be transformed in order that it can be presented to users in terms of a vocabulary and layout that is tailored for the users' needs. In fact, the transformations required could be extremely complex in terms of renaming, reordering, restricting, redividing, and recombining portions of the data as described by the conceptual schema.

4.2. The role of the conceptual level in videotex

As discussed in Section 3.2, the concept of schemas described separately from the instances has at most a limited applicability to standard videotex systems. As a result, it may seem at first that the Ansi/Sparc architecture has no relevance to the design of a videotex database system. However, to replace the concept of a schema, we introduce here the concept of a perception of data. Similarly to a schema, a perception is an abstraction of an instance; but unlike a schema, a perception includes an overview of

data content as well as data organization. A description of a user-oriented perception is what should be stored in a videotex data dictionary in order to provide the metaservice discussed in Section 3.2.

Consider, for example, the Department of Communication's demonstration Telidon database. The instance consists of the thousands of pages linked together in a particular tree structure with particular cross-links in place. A schema for this database has been shown in Figure 2b. A perception, however, would indicate that from the root page there are subtrees containing general information, information about Canada, information about the provinces, information provided by TVOntario, and so forth. The level of detail included in the perception is dependent on the expected user behaviour: there must, in fact, be a variety of detail to provide users with "several maps from very general ones to more specific ones, e.g., from galaxy to star system to planet" [Lochovsky 81, p. 25].

Similarly to providing several schemas for a traditional database, a videotex database will require the provision of several perceptions. It is apparent that a videotex database system, as any other database system, must provide a means for defining the actual storage representation for the data. It is also clear that videotex users will want to be presented with a view of data that is

distinct in general, from that representation (e.g., a logical tree of pages instead of a hash table assortment of pages). Thus there must be provision made for at least an internal and one or more external perceptions in videotex database systems.

In this section, we will consider the appropriateness of a third level of data description as embodied in a conceptual perception. As an example only, we present the following three levels of perceptions. One external perception of a videotex database may be a collection of pages, each accessible by means of keywords chosen from those listed on the page named KEYWORDS; the pages are perceived to be organized by subject matter. The conceptual peception may be a tree of pages in which each information provider is allocated one subtree; each page may be assigned one or more identifying keyword labels which will be automatically maintained by the system, as well as a numeric label indicating its position in the tree. Finally the internal perception might be a collection of B-trees of pages; one such B-tree, having integer keys of at most ten digits, is used for storing information pages about cars, another is used for pages supplied by American Express, still another with integer keys is used for action pages, and a fourth, with alphanumeric keys, is used for the keyword-to-numeric label translation. The data dictionary must store descriptions of these perceptions and allow for mappings between them.

How does a conceptual level, then, fit into a videotex database architecture in general? First, we notice that the system operator can be identified with the database administrator's role (managing the internal perception in the form of a set of stored pages) and information providers can be identified with the role of applications administrators (managing external perceptions as subsets of pages presented in forms suitable for end users). There may be additional applications administrators who manage pages not associated with individual information providers (e.g., system directory pages) or who provide views of the database in terms of integrated services composed of pages taken from many information providers (e.g., a travel agent service composed of integrated pages provided by airlines, train and bus companies, hotels, restaurants, oil companies, automobile clubs, banks, shop keepers, etc.). In fact, such a provision of external views derived from and based on other external views is included in the Ansi/Sparc proposal.

With this assignment of roles, what is the videotex counterpart to an enterprise administrator? The answer is a "master information provider", from whose vantage point all pages are seen as comprising a logical enterprise. The conceptual perception is then an integrated view of all the data available in the videotex system independent of the data's source or users.

Consider the database architecture in Figure 6 [cf. Bochmann 81, Figure 5]. In that framework, the end users access the database via user agents (either query or update agents) through which the appropriate information provider, metaservice, and/or actual database is discovered and activated. The query and update agents provide service directories (and perhaps some local parts of the database) to all users, and they form the entry points to the database for end users and information providers; they therefore embody the external perceptions. The page-oriented databases (together with any record-oriented databases, video and audio databases, and externally controlled databases) are accessed via database agents, whose directories embody the internal perception. Between these two classes of agents is a user-independent and database-independent model of the data that corresponds to a conceptual perception.

The advantage of such a conceptual level is the same as for traditional databases: each user agent and database agent can be altered independently, that is, without there being any modifications made to other agents. The only impact is to update the mapping from the conceptual perception to the perception managed by the altered agent.

Unlike in traditional corporate databases, the diversity of videotex data sources and uses is indicative of less centralized authority and control over the data con-

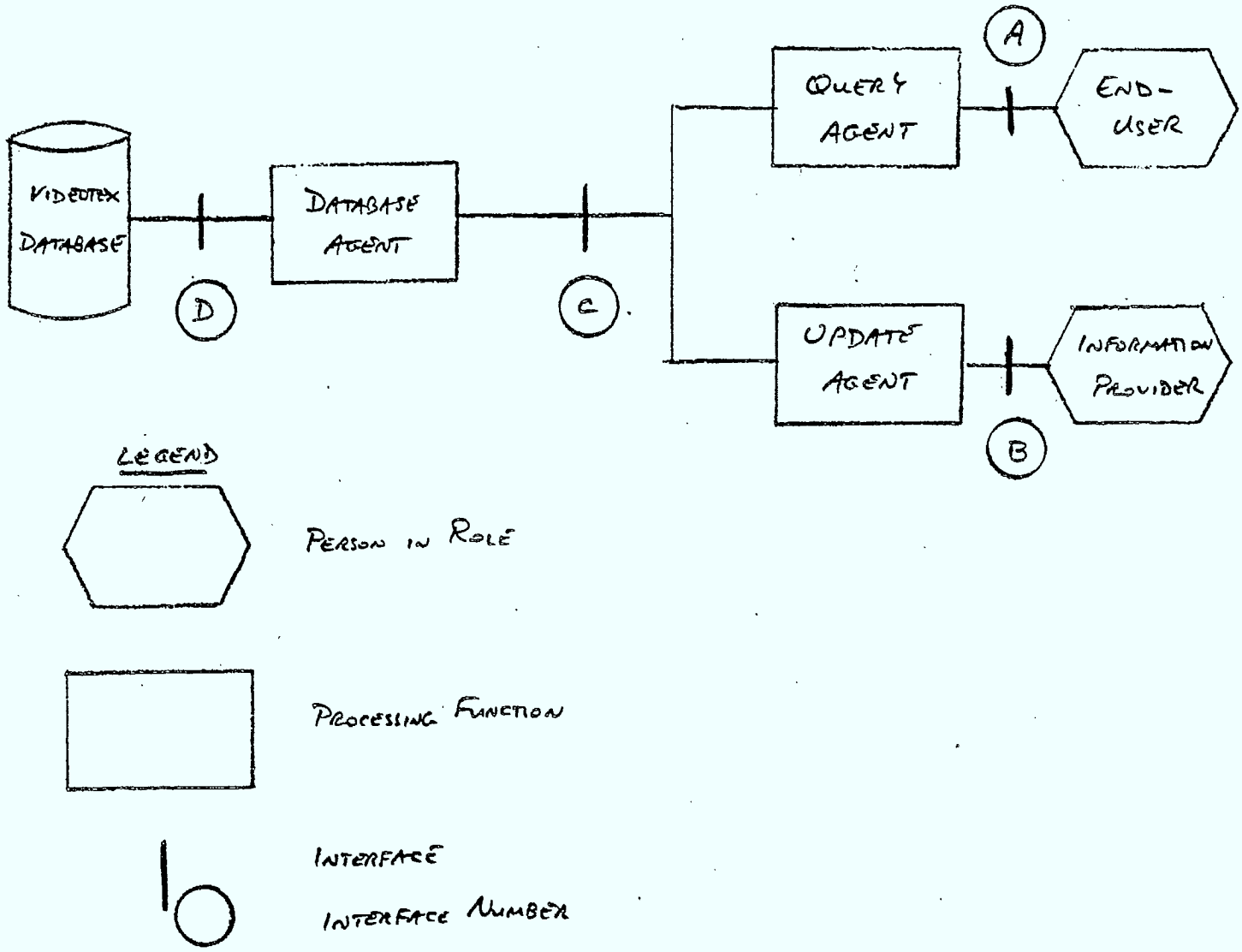


Figure 6: Simplified Videotex Database Architecture

tent. As a result, the videotex conceptual perception need not be the "universe of discourse" in terms of which all data in the other perceptions are described and from which the other perceptions must be derived. Instead it provides an overview of the data described in more detail in the other peceptions. The level of detail in the conceptual perception must be only enough to allow decisions to be made with respect to security or integrity constraints, cost-benefit analyses involved in providing additional services or allowing additional information providers, cost-benefit analyses in expanding or altering storage media, communications paths, and data distribution, and in maintaining the bridge from the external to the internal perceptions.

4.3. Some particular architecture interfaces

As mentioned in the introduction, the Ansi/Sparc architecture is defined in terms of forty-two database system interfaces (Figure 7). In this section, some of the interfaces will be described in slightly more detail.

Interfaces 1 and 2 provide the mechanism for recording aspects of the conceptual perception. Interface 3 provides the access to that information for information providers and system operators. Interfaces 4 and 5 are used for recording the particulars of the information providers' data definitions and for storing these in the directories controlling user agents. Interface 6 is the access route to

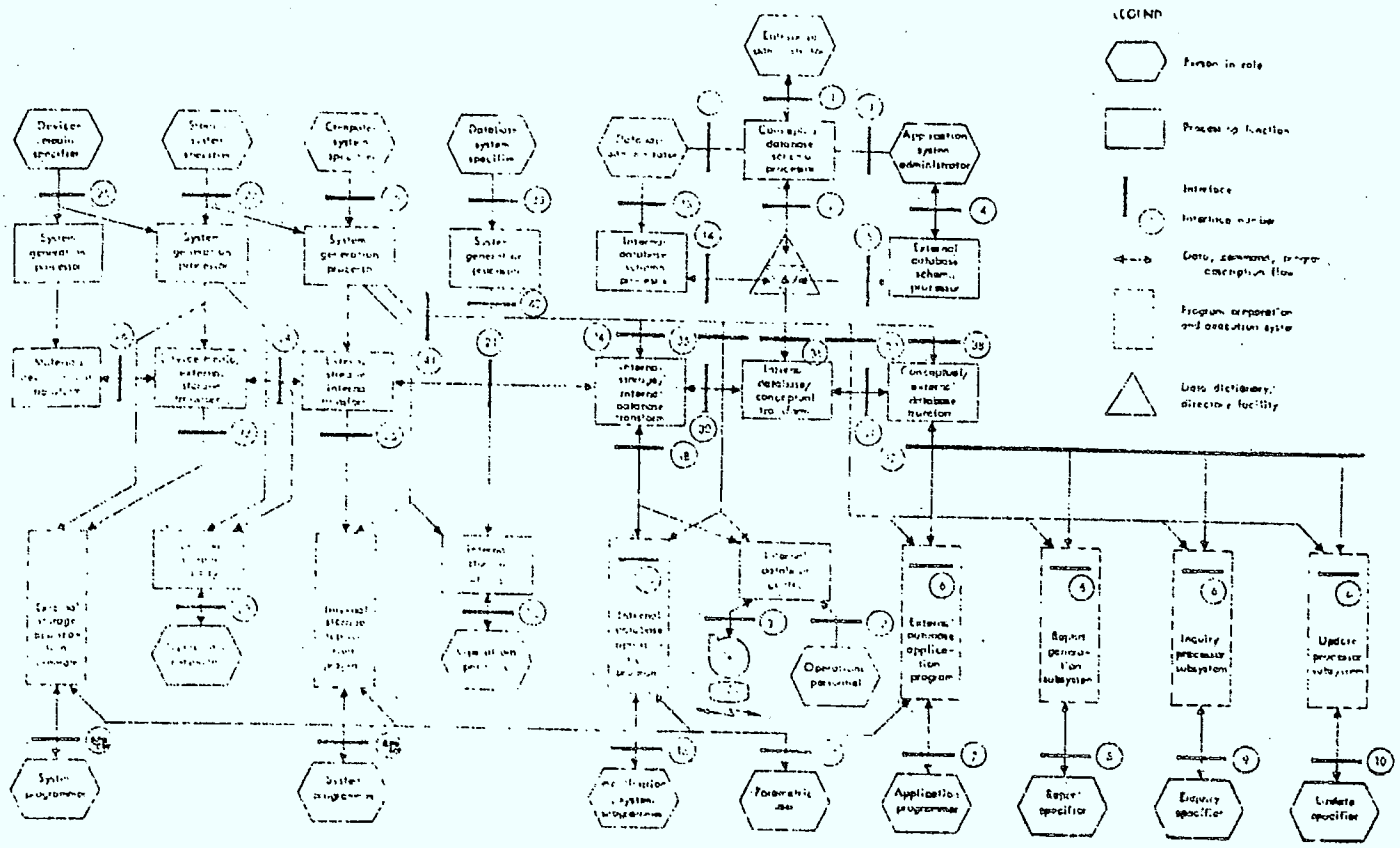


Figure 7: ANSI/SPARC ARCHITECTURE [Tormack 77]

those definitions for actual data manipulation and querying, the information providers generally using first interfaces 7, 8 and 10 and end users generally entering via interfaces 9 or 11. Interface 12 is used to ensure that every access to the data passes through security, integrity, and accounting control by routing requests via the conceptual perception.

The remaining thirty interfaces deal mainly with the internal operation of the database system and are typically under the control of the systems operator. The roles of these interfaces are essentially the same in a videotex system as in any traditional corporate database system. Of particular interest, perhaps, are interfaces 21, 30, 31 and 34-38 which link the user and database agents, i.e., effect the routing of data and commands from their specification in terms of an external perception to their internal storage description and back.

In concluding this section it should be remarked that the Ansi/Sparc architecture was designed as a basis for standardization discussions. It is therefore very general and need not be implemented exactly as described. In fact, the members of the committee did not expect every database system to have each interface present as an identifiable entity; rather they expected the functionality to be present everywhere. Thus in trying to implement an Ansi/Sparc-based

videotex system, it may be appropriate to combine several interfaces, duplicate some others, and perhaps even realize several in the hardware.

5. External database access

The idea underlying initial implementations of videotex was to provide access to a large database created and maintained especially for videotex. As the principle of videotex has become widespread and its technology more mature, it has become clear that the exclusive use of a database dedicated to videotex is too limited, inefficient and expensive. Large amounts of potentially useful information already exist in various databanks and services, which were designed independently of videotex; examples include bibliographic, medical and news databases. These so-called external databases may take any form, may run on any computing equipment under any operating system, and may have arbitrary security and integrity constraints defined on them. Automatic conversion of such databases to a uniform videotex format and structure (e.g., to Telidon) is generally very hard (or impossible) because of the great variations in display formats as well as in the internal data structures and models, search mechanisms, and interface languages.

Providing access to external databases is of great

economic interest to videotex users and external database operators, even if each external database has a different user interface. Such an access facility, which is currently receiving much attention, uses the videotex database machine as a gateway, i.e., a link to the external databases. Alternatively, the gateway function can be implemented on a processor that does not itself provide a dedicated videotex database, but rather is only a switch for a broad spectrum of services, as in the iNET system [CCG 81].

5.1. Compatibility between videotex and external databases

A videotex database and an external database, accessed through a gateway, are generally incompatible. Borrowing from the terminology of the ISO reference model for open system interconnections [Tanenbaum 81], we can identify three areas in which differences might occur: the application, presentation, and session levels.

At the application level are included differences in users' views of the data, as well as differences in the interface languages used to interact with the database. It is very difficult to mask these differences (i.e., to achieve compatibility at this level), unless the external database in question has been designed with compatibility in mind. For example, it would be foolish to consider providing an interface program to make the ORBIT retrieval system behave like a stored page videotex system [Ball 80a].

Consider next the presentation level. An external database may use character sets, codes, screen formats, line lengths, or graphics primitives other than those used by videotex databases and terminals. It is essential that some degree of compatibility be assured at this level, because it involves the terminal hardware (which cannot be changed), unlike the application level which involves humans capable of adapting themselves to different interfaces.

In general, the difficulty involved in achieving compatibility at the presentation level depends on the structure of the data contained in the external database. If that data consists of unformatted character strings (such as conventional text), then the required format conversion would consist merely in generating "carriage return" characters at appropriate intervals (e.g., forty characters per line for Telidon). This can be done either in a gateway computer, or in the terminal itself. New paragraphs can easily be detected and handled similarly. An example of this solution is found in the Cable News Processor (see Section 5.3).

Alternatively, consider external databases that contain formatted text, but neither figures nor diagrams. A typical example would be text with tables of numbers arranged in rows and columns. Adopting a solution as above, i.e., inserting carriage returns, might be the only pos-

siblity; for some applications this would yield low quality, but acceptable, displays. Other applications would not be conducive to such naive reformatting, and therefore it would be better to use higher resolution displays. For example, such a modified TV-based videotex terminal is being developed by Electrohome to allow eighty characters per line (the standard for conventioanl alphanumeric terminals) to which most traditional database output conforms.

Finally an external database may contain figures, diagrams or similar graphical information. If this information is coded using graphical primitives, then translation might be possible, although perhaps costly in terms of processor utilization if the primitives are too unlike Telidon's PDIs. Alternatively, if the graphical information is coded alpha-photographically or if it exists in analog form (e.g., on videodisk), then appropriate display would only be possible with the addition of suitable hardware attached to the videotex terminal.

Consider finally the session level, which usually deals with functions such as logon and user identification procedures, accounting, transaction integrity, and the control of an ongoing session. In using a gateway, such as iNET or Bildschirmtext, most of these functions can be made invisible to the user, as they are typically handled by programs in the gateway computer. This level should not

cause insurmountable problems for achieving compatibility between videotex and external databases.

Before concluding, we note that areas corresponding to the lower levels of the ISO reference model are practically invisible to the user. Still, of course, mutually compatible protocols must be used by the videotex and external database computers.

5.2. Alternatives for providing interfaces

There are several possibilities for providing access to external databases. In this section, we consider the impact of these interface strategies on compatibility as discussed above. First, the external database may be an independent database designed (typically) before videotex has become known and accepted. Because the database software contains no component designed to interface with videotex, the videotex gateway must appear as just another of its users (Figure 8). The gateway processor, therefore, generally must contain some adaptation module which performs conversions necessary for terminal compatibility at the presentation level. This module may also perform some functions associated with the session level. As mentioned above, typically there is no compatibility in the application area: thus the user must use query procedures appropriate for the external database.

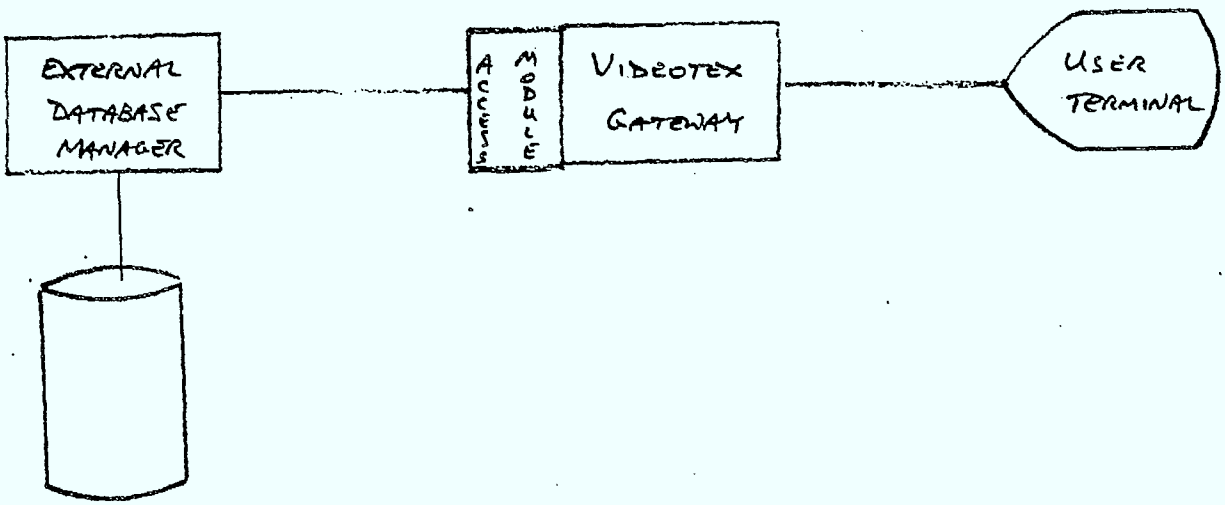


Figure 8: VIDEOTEX - INDEPENDENT DATABASE ACCESS

Alternatively, even if the external database is an independent database as above, the adaptation module may be able to reside in the external computer, rather than merely in the gateway (Figure 9). in Figure 9. The two adaptation modules implement an extended presentation level protocol which, in addition to providing a minimum of necessary format conversions, may also provide for functions such as presentation and handling of data acquisition pages as in Bildschirmtext [Otto81a]. Such a protocol will likely also handle session level functions (e.g., logon). A certain level of compatibility in the application area can be realized as well (e.g., the limited use of menus with numerical selections); however this applies more to transactional applications (which can be done with a limited number of pre-programmed action pages) than to traditional database retrieval.

Finally, the external database can be a third-party (externally controlled) videotex database. If both databases are of the same type (e.g., both Telidon), then complete compatibility can be easily achieved at all three levels. If, however, the databases are merely similar in structure (e.g., one is Prestel based and the other Telidon-based), then application and presentation level compatibility are possible (but not trivial) to achieve. To date, we know of no attempt to make Prestel and Telidon compatible at the application level. On the other hand, the

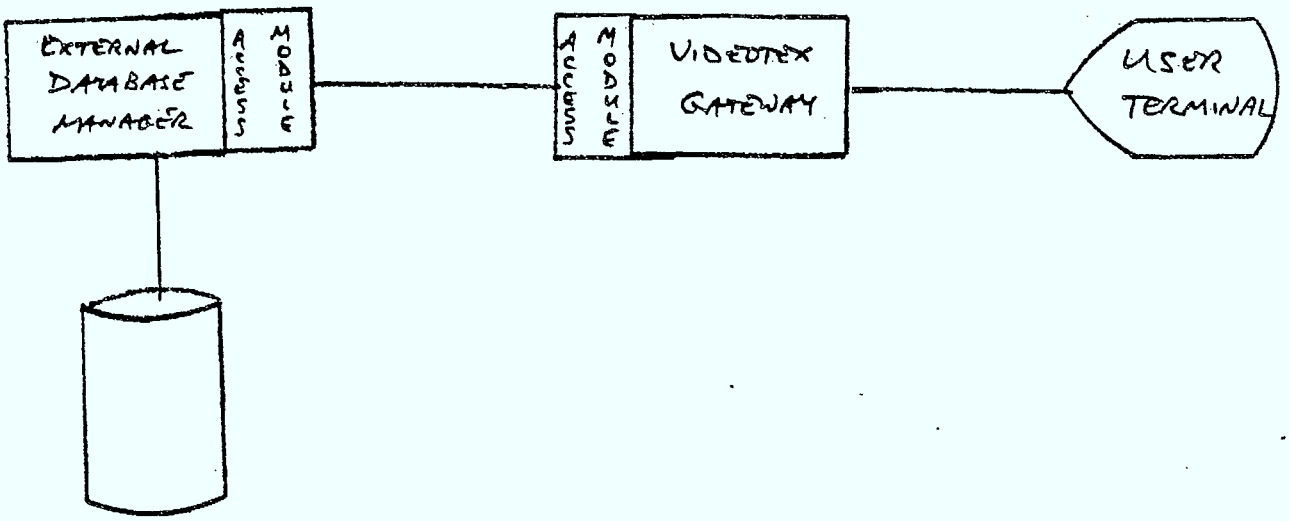


Figure 9: ADAPTED EXTERNAL DATABASE ACCESS

newly announced AT&T standard [Bell 81] makes the problem of compatibility at the presentation level potentially simpler to solve, especially for Telidon/Teletel interfaces.

5.3. Current developments

Certainly the most important development to date in providing access to external databases is the Bildschirmtext network in Germany. This system is based on the premise that access to external databases (and computers in general) will be a fundamental function in future videotext systems and not an exceptional, add-on feature. The backbone of this system consists of a number of "local Bildschirmtext computers", each servicing a number of users. Each local computer provides a basic videotex database service, a menu of available external computers, and a set of gateway pages. Each gateway page (or rather the program associated with it) controls the call, logon, and transfer of information between the external computer and the user terminal. Two kinds of functions are available: database access and data acquisition. The latter serves to collect and transmit user input for interactive applications. The most important problem of compatibility is solved by the inclusion of a special "external computer access protocol", situated at the session and presentation levels. system architecture. This protocol is implemented on each external computer, and is analogous to the adaptation module shown in

Figure 9. The protocol is said to be simple and efficient, and is supposed to be replaced by a more complete international standard if and when such a standard exists [Otto 81b].

As a second example, the Cable News Processor [MD 81] is a microprocessor-based system that creates a "pictorial multicoloured news service" for Telidon terminals on cable TV lines. The unit automatically captures stories from a conventional wire service, reformats them, and segregates the information by category. Alternative and additional information can also be acquired from information providers. A similar service also exists in Great Britain for Prestel.

A third example of a gateway is the recently announced iNET trial to be conducted by TransCanada Telephone Systems next year. The iNET system uses the interface strategy shown in Figure 8. Essentially it serves as a user-friendly interface to Datapac, Bell Canada's public data network. An iNET user may access any of a large number of independent external computers through menu selection; the gateway then automatically performs sign-on and the execution of an initialization profile (e.g., to select a particular first page in a videotex system). When a user signals a (standard) sign-off to iNET, the gateway invokes the appropriate sign-off procedure for the external computer.

The iNET system also maintains the accounts for its users in order that centralized billing is available.

6. Conclusions and recommendations

This study of videotex database systems has shown that, although there are many differences between videotex and traditional corporate databases, several of the principles and practices of corporate database systems can be fruitfully applied to videotex environments. The principal difference stems from the fact that videotex relies on a diversity of information sources and uses; thus, the role of a schema is far less significant than for traditional systems. A second difference involves videotex's proximity to transaction processing; as a result, the role of an integrated form-filling facility becomes more significant than in most corporate systems.

Much of the functionality of traditional database systems must be present in a videotex environment. In particular, the functional components of the systems will be very similar, albeit adapted in videotex systems chiefly for economic reasons. Furthermore, the architecture of the systems should also be very much the same.

As a result of this study, we recommend that the following actions be taken:

. The applicability of types to defining "page clusters" should be examined in more detail. Can such clusters be treated, for example, as abstact data types in order that information providers and end users can deal with a collection of pages meaningfully? This approach may well lead to a data definition language that could automate the generation of index (i.e., routing) pages for videotex.

. Specification techniques for data transformations should be investigated in order to determine their applicability to automating database loading and modification. Such a facility would simplify stored page construction (as for the demonstration connection to Associated Press's wireservice), dynamic page assembly from record-oriented databases, and dynamic updating of databases accessed through videotex forms.

. A three-level videotex database architecture should be examined in more detail. First of all, a language for the specification of perceptions should be developed (based on an examination of such a language's features). Secondly, the Ansi/Sparc interfaces should be individually studied for their role in a videotex system and their suitability for standardization in videotex.

. Finally, access to external computers by Telidon users should be pursued as a design goal in future systems. The facility has great potential economic impact on the com-

munity of external computer operators as well as on the market potential for Telidon terminals (as a cheap way to access a variety of services). The Bildschirmtext external access protocol should be studied in detail and this or a similar protocol should be considered for Canadian use and later as an international standard.

References

[Ansi 75] Ansi/X3/Sparc, "Interim report from the study group on database management systems", ACM FDT 7, 2 (December 1975) 140 pp.

[Bachman 69] C.W.Bachman, "Data structure diagrams", ACM's Data Base 1, 2 (Summer 1969).

[Ball 80a] A.J.S.Ball, "Adaptation of existing library databases to videotex", Proc. First Montreal Workshop on Videotex Technology, Tech. Rept., Dept. d'informatique, Univ. de Montreal, June 1980.

[Ball 80b] A.J.S.Ball, G.V.Bochmann, and J.Gecsei, "Videotex networks", IEEE Computer 13, 12 (December 1980) 8-14.

[Ball 81] A.J.S.Ball and J.Gecsei, "User interfaces for future videotex systems", Dept. d'informatique, Univ. de Montreal, unpublished, June 1981.

[Bell 1981] Bell System, "Videotex standard presentation level protocol", AT&T, May 1981.

[Bochmann 81] G.V.Bochmann, "Overview of protocols in distributed videotex systems", Dept. d'informatique, Univ. de Montreal, unpublished, May 1981.

[CCG 81] Computer Comunications Group, "iNET: Gateway to the information marketplace", TransCanada Telephone System, unpublished publicity folder, May 1981.

[Codasyl 71] Data Base Task Group of Codasyl Programming Language Committee, Report (available from ACM), April 1971.

[Comer 79] D.Comer, "The ubiquitous B-tree", ACM Computing Surveys 11, 2 (June 1979) 121-137.

[Gottlieb 77] D.Gottlieb, S.A.Hagerth, P.G.H.Lehot, and H.S.Rabinowitz, "A classification of compression schemes and their usefulness for a large data processing center", AFIPS Nat. Comp. Conf. 44 (1975) 453-458.

[Gray 79] J.C.Gray, "Notes on Database Operating Systems", IBM Res. Rept. RJ2188, San Jose (Feb. 1978) 91 pp.

[Henriot 79] A.Henriot and J.Yclon, "Langage de programmation des bases de donnees Star", CCETT Note Technique RSI/41/443/79, Rennes, France, December 1979, 39 pp.

[Infotech 77] Infotech, State-of-the-Art Report on Performance Modelling and Prediction, 1977.

[Kerschberg 76] L.Kerschberg, A.Klug, and D.Tsichritsis, "A taxonomy of data models", Systems for Large Data Bases, North-Holland, 1976, 43-64.

[Le Moign 80] D.Le Moign, "Presentation de Star", CCETT Note Technique RSI/NT/23/17/80, Rennes, France, March 1980, 13 pp.

[Lochovsky 81] F.H.Lochovsky and D.C.Tsichritzis, "Interactive query languages for external databases", Tech. Rept. CSRG, Univ. of Toronto, March 1981, 48 pp.

[McLeod 81] D.McLeod and J.M.Smith, "abstraction in databases", Proc. of the Workshop on Data Abstraction, Databases, and Conceptual Modelling, Sigmod Record 11, 2 (February 1981) 19-25.

[MD 81] Macdonald-Dettwiler, "Cable News Processor", unpublished publicity leaflet, May 1981.

[Mealy 67] G.H.Mealy, "Another look at data", Proc. AFIPS 31, (FJCC 1967) 525-534.

[Otto 81a] J.Otto, "Access to private data bases: an essential supplement to the interactive videotex system concept", unpublished, 1981, 18 pp.

[Otto 81b] J. Otto, private communication.

[RSI 79] Relational Software, Inc., "Oracle introduction", Menlo Park,, 1979.

[Senko 73] M.E.Senko, E.B.Altman, M.M.Astrahan, and P.L.Fehder, "Data structures and accessing in data-base systems", IBM Sys. J. 12, 1 (1973) 1-93.

[Sibley 76] E.H.Sibley (ed.), "Special issue: data-base management systems", ACM Computer Surveys 8, 1 (March 1976) 151 pp.

[Tanenbaum 81] A.S.Tanenbaum, Computer Networks, Prentice-Hall, 1981.

[Taylor 79] K.H.Taylor, "On-line business databases and viewdata", Proc. Online 79, Online Conf. Ltd., London, UK, 1979, 273-282.

[Tompa 77] F.W.Tompa, "Data structure design", in Data Structures, Pattern Recognition, and Computer Graphics, A.Kliner, T.Kunii, and K.S.Fu (eds.), Academic Press, 1977, 1-33.

[Tompa 81] F.W.Tompa, J.Gecsei, and G.V.Bochmann, "Data structuring facilities for interactive videotex systems", IEEE Computer 14 (to appear 1981).

[Wiederhold 77] G.Wiederhold, Database Design, McGraw-Hill, 1977, 658 pp.

[Yormack 77] B. Yormack, "The Ansi/X3/Sparc/SGDBMS architecture", in The Ansi/Sparc Model, D.A.Jardine (ed.), North Holland, 1977, 1-21.

[Zloof 75] M.M.Zloof, "Query By Example: the invocation and definition of tables and forms", Proc. VLDB (Sept. 1975).