LAN:DataCore

Introduction

What Is LAN:DataCore™ And Why Use It?

LAN:DataCore is a powerful relational database development tool for microcomputer local area networks. With it, you can write your software applications in a fraction of the time it would take to write them from scratch.

The LAN:DataCore library includes a full set of programmatically callable MS-DOS procedures to create databases and to store and retrieve information from them. This means your users can be completely shielded from the internal details of the Database Management System—they only need to learn your application. In addition, you can save development effort by supplying LAN:DataStore to handle your information retrieval or database creation needs.

The increasing popularity of local area networks opens an enormous market opportunity for software applications. Unfortunately, the software development process is often tedious and time consuming. With LAN:DataCore, you only need to write the user interface portion of your application. The information storage and retrieval is handled by LAN:DataCore.

LAN:DataCore handles all the concurrent access coordination required for the network, so your application doesn't have to. Up to 16 users can read, write, update, and delete information from the same database at the same time. The passive record locking mechanism ensures database integrity and is competely automatic — you don't have to write complex synchronization mechanisms in your programs. Furthermore, LAN:DataCore doesn't require a dedicated machine as a "database server." This saves you money and eliminates a potential bottleneck.

LAN:DataCore provides multi-user security and privacy controls, so you can reap the benefits of a centralized database without compromising sensitive information. Different users can be granted access to certain fields within the records or to certain records within the database. Selection criteria can be used to provide a subschema facility — one database might "look" different to different users. LAN:DataCore can also be used to grant, or restrict, read, write, update, and delete authority for each user.

LAN:DataCore efficiently handles databases large enough for most business applications — up to 16 Mbytes of information per database. Yet, the large database capacity does not compromise the performance. The high performance index organization allows any record to be located in a fraction of a second, while eliminating the need for time consuming sorts or index restructuring.

How To Use This Manual

Before writing any programs, you should review the LAN:DataCore structure and security features as described in section II. You can then determine the best database structure for your application.

Section III explains how to write a program with LAN:DataCore. It provides information about the interfaces and the linking procedure.

Sections IV and V are reference sections describing how to use each of the LAN:DataCore intrinsics. These sections provide descriptions of the calling conventions, the parameters, the modes, and the actions of each procedure. The errors possible for a procedure are also explained there.

There is a utility provided with LAN:DataCore — CLEARUSR. This is used to close a database for a user or to unlock a semaphore in the event of a machine or software failure. Section VII explains when this is necessary and how to use the utility.

Appendix A contains a listing of a sample program that uses a LAN:DataCore database. You can look at this program to see how the intrinsics are called and how the parameters are set up, and use it to verify that your application is working.

Appendix B provides an overview of the database internal operation, including a description of the storage requirements, the internal structure, and the limits of a database.

Finally, Appendix C is a summary of the errors. Additional information is provided with each of the intrinsics.

What's In The LAN:DataCore Package?

You will find the following files on your LAN:DataCore floppy diskettes. You should copy the .INT and .LIB files to your hard disk:

Interfaces to include in your Pascal program:

PINDBDEC.INT	Database declarations
PINDEFIN.INT	Interfaces for database definition
PINOPCL.INT	Interfaces for Open, Close processing
PINIOROU.INT	Interfaces for I/O routines

Library to link in with the object code file(s) for your program:

DCORE.LIB	LAN:DataCore intrinsics.

Source code for sample test programs and a test program:

PINTEST.PAS	Sample LAN:DataCore test program source
PINTEST.EXE	Test program execution file
DBCREATE.PAS	Test program which creates a database (source)
DBUSER.PAS	Test program which creates users (source)
DBIO.PAS	Test program which reads/writes (source)
PTLCRT.INT	Interface file for DB** applications

BAT files to link LAN:DataCore with your application:

XINLINK.BAT	Batch file to link LAN:DataCore application
XOBLINK.LNK	Link parameters

CLEARUSR utility to force a database closed:

CLEARUSR.EXE	Clear Users Utility

Hardware And Software Requirements

LAN:DataCore operates on an IBM PC or compatible or a Texas Instruments Professional under MS-DOS version 1.1 or 2.0. The LAN:DataCore library requires approximately 70K of memory. In addition, the PC-DOS PASCAL library requires an additional 25K. Add the size of your application and the MS-DOS operating system to determine the memory required of your computer.

The multi-user version of LAN:DataCore requires one of the following local area networks:

—Corvus OmniNet.

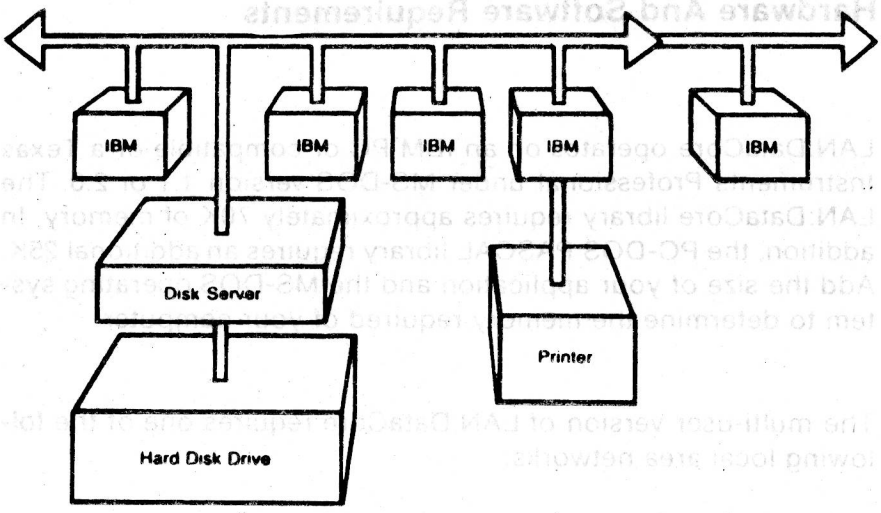
—3Com EtherSeries local area network.

—Novell ShareNet or Gateway G-Net.

—Davong Systems Multi-Link.

—Orchid Technology, Santa Clara Systems or A.S.T. PC-Net.

LAN:DataCore does not require a dedicated or central computer as a "database server". Instead, each machine accesses the disk volume with the database directly. This saves you money and eliminates a potential bottleneck. As many as 16 machines may open a common database open at any time.



LAN:DataCore Topology

What is a LAN:DataCore Database?

Database

A database is a collection of related information. A relational database can be thought of as a table. The rows of the table (records) contain a description of an object or event. The columns of the table reflect certain attributes of each object or event.

For example, a personnel department might need to store information for each of the company's employees. A row of the table would be allocated for each employee, while the columns would contain the different fields that must be known for the employees:

Empl#	Name	Address	City	State
1046	Buck Gee	1702 Main St.	Alviso	CA
1076	Eileen Rose	46 Williams St.	Dallas	TX
2037	Gary Kwok	233 Willow Ave.	Miami	FL
2071	Mike Hartstein	748 Trent Blvd.	Clark	NJ
2307	Chris Crump	151 Brain St.	Hilo	HI
6018	Mike Lipsie	33 Sunset Terr.	Oakland	CA

Records

A database is a collection of records. Each record is a set of information for an event or object. For instance, a personnel database might have many records, each containing information on the employees of a company. An order database, on the other hand, might have records each of which correlates to an order. These correspond to rows of information in a table. The example above shows a database with 6 records.

Fields

A field corresponds to an individual piece of information or a column in a table. For instance, the personnel database might contain five fields — an employee number, name, address, city, and state.

LAN:DataCore allows several different types of fields. Data can be an integer (16 bit two's complement number), a real (32 bit floating point number), a character array (1 to 4K bytes), a name (1 to 4K bytes represented as "First-Name Last-Name," i.e. "John Smith," but sorted by the last name, a date (MS-DOS format — integer year, two bytes indicating the month and day), or a packed decimal number (8087 format — 10 bytes, 18 digits).

LAN:DataCore is a field level access Database Management System. Therefore, your program can be independent of the database structure. When a record is read, only those fields specifically requested are returned. This means working programs will continue to work when new fields are added to or deleted from the database.

Keys

Up to 16 fields may be designated as keys. Keys allow records to be retrieved directly and in sequential (sorted) order. LAN:DataCore's sophisticated B+ tree key algorithm provides optimal performance for record searches.

Since many fields may be designated as keys, the records may be retrieved in several sequences. For instance, if the employee number, name, city, and zip were keys, you could retrieve the employee records in order of employee number, name, city, or zip.

Unlike other less sophisticated databases, LAN:DataCore dynamically maintains the key structures. Each time a record is inserted, deleted, or modified, the key structures are updated. This means you never have to do any sorts or index rebuilds — the records are always available in sorted order.

As many as 16 fields may be designated as keys, as long as each is 242 bytes or less in length. All field types may be used as a key. The keys need not be the first fields in the record — any field may be designated as a key.

LAN:DataCore supports both unique and duplicate keys. Unique keys are used when the value of a field cannot appear in more than one record. For instance, an employee number must be unique — duplicates are not allowed. If you designate a key as being unique, LAN:DataCore will prevent duplicate entries of it from occuring.

Duplicate keys permit two or more records to contain the same value for a field. A state (i.e. California, Nevada, etc.) is a good example of a duplicate key. It is very likely several employees live in the same state — you certainly don't want the database to reject them.

The personnel database example contains three keys. The list of records which appears above shows the order of the records when retrieved in order of the Empl# key. The database appears in the following order when the records are retrieved in order of the Name key:

Empl#	Name	Address	City	State
2307	Chris Crump	151 Brain St.	Hilo	HI
1046	Buck Gee	1702 Main St.	Alviso	CA
2071	Mike Hartstein	748 Trent Blvd.	Clark	NJ
2037	Gary Kwok	233 Willow Ave.	Miami	FL
6018	Mike Lipsie	33 Sunset Terr.	Oakland	CA
1076	Eileen Rose	46 Williams St.	Dallas	TX

You may also read the records in order of the other key — State. The database would then appear as:

Empl#	Name	Address	City	State
1046	Buck Gee	1702 Main St.	Alviso	CA
6018	Mike Lipsie	33 Sunset Terr.	Oakland	CA
2037	Gary Kwok	233 Willow Ave.	Miami	FL
2307	Chris Crump	151 Brain St.	Hilo	HI
2071	Miké Hartstein	748 Trent Blvd.	Clark	NJ
1076	Eileen Rose	46 Williams St.	Dallas	TX

Notice that in each case, the database consists of the same information. Only the order of the records varies. This means you can print reports or search through the database in one of three orders — Empl#, Name, or State. Other fields can be designated as keys if necessary.

The example also illustrates the difference between unique and duplicate keys. Since the company never assigns an Empl# to more than one person, the Empl# key should be unique. Although unlikely, it is possible that the company could have two employees with the same name. And it is very likely that two employees live in the same state. Those two fields should be duplicate keys.

When determining which fields to designate as keys, keep in mind that B+ trees require substantial storage and increase the time required to insert or delete a record. Also, fields with very few possible values (such as yes/no questions) are not good candidates for keys.

Selection Criteria

LAN:DataCore lets you specify selection criteria to limit the records a user has access to, or to simplify the retrieval of a definable class of records.

When the database creator first defines a database and its users, a security and privacy matrix is tagged to each user. This matrix includes selection criteria which can limit the records the user has access to to some subset. Consider the following database:

Empl#	Name	Address	City	State
1046	Buck Gee	1702 Main St.	Alviso	CA
6018	Mike Lipsie	33 Sunset Terr.	Oakland	CA
2037	Gary Kwok	233 Willow Ave.	Miami	FL
2307	Chris Crump	151 Brain St.	Hilo	HI
2071	Mike Hartstein	748 Trent Blvd.	Clark	NJ
1016	Eileen Rose	46 Williams St.	Dallas	TX

One user might be limited to seeing only those personnel records for employees that live in California. The database would use selection criteria to permit access only to records where the State field has a value of **CA**.

The database would look as though it only had the following contents:

Empl#	Name	Address	City	State
1046	Buck Gee	1702 Main St.	Alviso	CA
6018	Mike Lipsie	33 Sunset Terr.	Oakland	CA

Another user might have access to employees that live in Florida and have an employee number that is greater than 2000.

The database will look like:

Empl#	Name	Address	City	State
2037	Gary Kwok	233 Willow Ave.	Miami	FL

Selection criteria may also be supplied to the **DBREAD** intrinsic to achieve the same result — to make the database appear as though it only contains some certain subset of the records. Thus, a program to produce a report about only the employees with a zip code of 95124 would not need to include code to select the appropriate records. Instead, LAN:DataCore could perform the selection internally.

Selection criteria is composed of a relation and value for any or all of the fields. The relation may be any of the comparison relations (less than, less than or equal to, greater than, greater than or equal to, equal to, or not equal to), or a prefix or suffix relation for character strings and names (for names, the prefix relation applies to the last name).

User Security Matrix

In addition to selection criteria to limit the records a user has access to, the creator may also limit access to certain fields within a record and limit the operations a user can perform on the database.

When defining a user, the creator specifies which fields the user has access to. For example, one user might have access to all fields in the personnel database while another might only have access to their name and employee number.

So, the database creator could decide that for a certain user the database should look like:

Empl#	Name
1046	Buck Gee
6018	Mike Lipsie
2037	Gary Kwok
2307	Chris Crump
2071	Mike Hartstein
1076	Eileen Rose

The creator may also grant or deny Read, Write, Update, and Delete authority to each user. So a clerk might be allowed to see some information but not change it.

Audit Trails

LAN:DataCore provides a unique audit trail feature to track the last user that added or modified each record. When the database creator reads each record from a database, an audit parameter is set to the user number, time and date for the last time the record was changed. Thus, the creator can determine, for example, who changed the salary of an employee or who entered the large order.

This information is available only to the person who created the database. It is not returned when other users open the database.

Writing A Program With LAN:DataCore

Calling The Intrinsics

LAN:DataCore provides a set of intrinsics that can be called from application programs to access a database. Several PASCAL INTERFACE files have been provided which define these intrinsics. These are included in the application program compilation. The compiled application program should then be linked along with the object code files from the LAN:DataCore package. Although the examples in this manual and the interface files are in Pascal, applications may be written in other langauges compatible with the MS-DOS linker and Pascal calling conventions.

A set of FORTRAN procedures have also been provided. These procedures have the same parameters as their PASCAL counterparts. Appendix A provides an example of a FORTRAN program which calls LAN:DataCore. Your application programs must call the LAN:DataCore intrinsics in the following order:

1. Call DBCREATE to create a new database.

2. Call DBBUFSIZE to determine the memory size needed for the buffer pool.

3. Call DBOPEN to open the database.

4. Call other LAN:DataCore intrinsics to write records to the database, update, delete, and read records.

LAN:DataCore requires memory for internal buffers. This memory must be allocated by the application program. Before calling the LAN:DataCore intrinsics, you should allocate two memory arrays. The first is called the DBCB (Database Control Block). This is a 98 word array which is initialized by DBOPEN and which must be passed into each LAN:DataCore intrinsic. Before calling DBOPEN, the first word of the DBCB should be set to the address of the other memory array, and the second word should be set to the size of the other memory array.

This second memory array is used internally for buffers. The size of this buffer may be determined before opening the database by calling the DBBUFSIZE intrinsic. This buffer should never be modified by the application program. There is a DEBUG parameter on DBOPEN. This parameter, when set to true, checks that the buffer hasn't accidentally been modified between database intrinsics. You should set this parameter to true while debugging your application, then change it to false to increase the database performance when your application goes into production.

Some intrinsics have PASCAL record structures to describe certain data. This includes the item descriptions for a DBCREATE, the user description for a DBDEFUSER, the selection criteria for a DBREAD, the DBCB before a DBOPEN, and several DBINFO modes. These records are predefined in the INTERFACE files, and you may use them.

LAN:DataCore is a general database package, and it permits you to have between 1 and 512 fields in each database, with records ranging in size from 1 to 16K bytes. Because this range is so large, you cannot declare all buffers for all databases with the same size — you will want your buffer to be only large enough to hold the records of the database you are using. Unfortunately, PASCAL was not well designed to handle variable size arrays.

To accommodate these variable size arrays, the intrinsics are designed to accept pointers to buffers. You must declare your array with the NoRange (0..0) range when passing the parameter into LAN:DataCore. Within your program, however, you may declare an array with the appropriate bounds.

There are several sample programs which illustrate these techniques for you to look at. PINTEST.PAS is a test program that simply asks you for parameters and calls the LAN:DataCore intrinsics.

If you are creating a database through LAN:DataCore and intend to use it through the LAN:DataStore package, you must correctly set the following fields:

CRPage — The form number, between 1 and 15, where the field is to be displayed.

CRCol — The column numbers, between 0 and 79, where the field name begins.