SAA distributed file access to the CICS environment

by K. Deinhart

IBM's Customer Information Control System (CICS) is the leading product family in the on-line transaction processing (OLTP) market. OLTP systems are being used by many enterprises to implement their daily business processes and manage operational data such as accounts, inventories, and orders. CICS/Distributed Data Management (CICS/DDM) implements the distributed file function of Systems Application Architecture® (SAA™) Common Communications Support in the CICS environment on Multiple Virtual Storage (MVS) and Virtual Storage Extended (VSE) operating systems. Providing a DDM target server, CICS/DDM implements IBM's SAA protocol for access to distributed data, which are exploited by the SAA Common Programming Interface. CICS/DDM allows applications and their users to access and share the data managed through the OLTP environment provided by CICS.

On-line transaction processing (OLTP) systems like the Customer Information Control System (CICS) are being used by enterprises of all sizes across industries as the basis for their key business processes. These systems capture and manage essential business information.

By implementing a target server in compliance with IBM's Distributed Data Management (DDM) architecture,[1-3] CICS/DDM provides the distributed file function of Systems Application Architecture* (SAA*) Common Communications Support (CCS)[4,5] in the CICS environment. It adds file access connectivity to the family of CICS products by providing access to CICS data from other systems through an open IBM protocol. This protocol has already been adopted by operating systems inside and outside of IBM and is currently being implemented in the SAA environment.

This paper focuses on the aspects of sharing CICS data stored under Multiple Virtual Storage (MVS) or Virtual Storage Extended (VSE) operating systems within a network configuration that has implemented DDM. It reviews major functions of CICS, which is IBM's flagship for OLTP and the basic subsystem available in most VSE systems. The functions and values of CICS are put into the context of SAA file I/O operations, thus identifying the impacts of connecting interactive single-user systems, such as the Conversational Monitor System (CMS), Time Sharing Option/Extended (TSO/E), Personal Computer Disk Operating System (PC-DOS), Operating System/2* (OS/2*), and OLTP environments, and outlining some similarities and differences between those systems. That is, CICS is used as an example for an OLTP system in order to identify in general the specific aspects of accessing shared data stored in OLTP systems.

The importance of a record access method in a distributed network that includes OLTP systems is explained. Finally, the implementation of CICS/DDM and its relevance to source system applications are discussed, highlighting the benefits of a DDM implementation to a user.

CICS and DDM: Benefits and history

Today, many enterprises have installed a variety of computers ranging in size from small to medium to large, many of which run different operating systems for different purposes. Some of these are even from different manufacturers. Along with this variety in computer systems comes a wide range of data management systems that have unique file organizations and file access methods. Sometimes there are even differences between subsystems on the same computer due to the fact that different requirements have to be fulfilled. For example, OLTP systems such as CICS and the Information Management System (IMS) utilize different data management I/O services than do interactive systems such as TSO and CMS.

When two different systems are connected to exchange data, communication programs have to be supplied. These programs are often specific to the application functions needed and the environment connected, so that a change in user requirements or changes in the hardware and software environment often require the programs to be modified or new programs to be written. When the communications network is complex, such as when several systems of different types need to be connected, the task of maintaining these communication programs can be considerable.

DDM is an open data management architecture that defines an SAA CCS protocol for data interchange between systems that may have different data management systems. It is open in so far as it is published, provides compatibility across different architectural levels, and has already been adopted by different file systems inside and outside IBM.

Communication is achieved through DDM source and target servers that exchange DDM data streams over logical unit (LU) 6.2 pipes: The source servers ("clients") run on the system of the application or end user requesting access to data; the target servers ("servers") run on the system where the data reside. With DDM, it does not matter whether two systems store, manage, or process data differently; once these systems are connected, an application program running on one system can gain access to data stored in the other system, the same as if the data were stored locally.

User and programmer productivity. For an application programmer the advantage of using DDM is to be able to write application programs that access data without needing to determine where the data actually reside; this extends the life and the value of an application. Functions written for access to a specific target system are easily available for other systems that have implemented DDM. The benefit to the end user is not having to learn different commands or services to access remote data from those needed to access local data. DDM does the necessary work to locate the data.

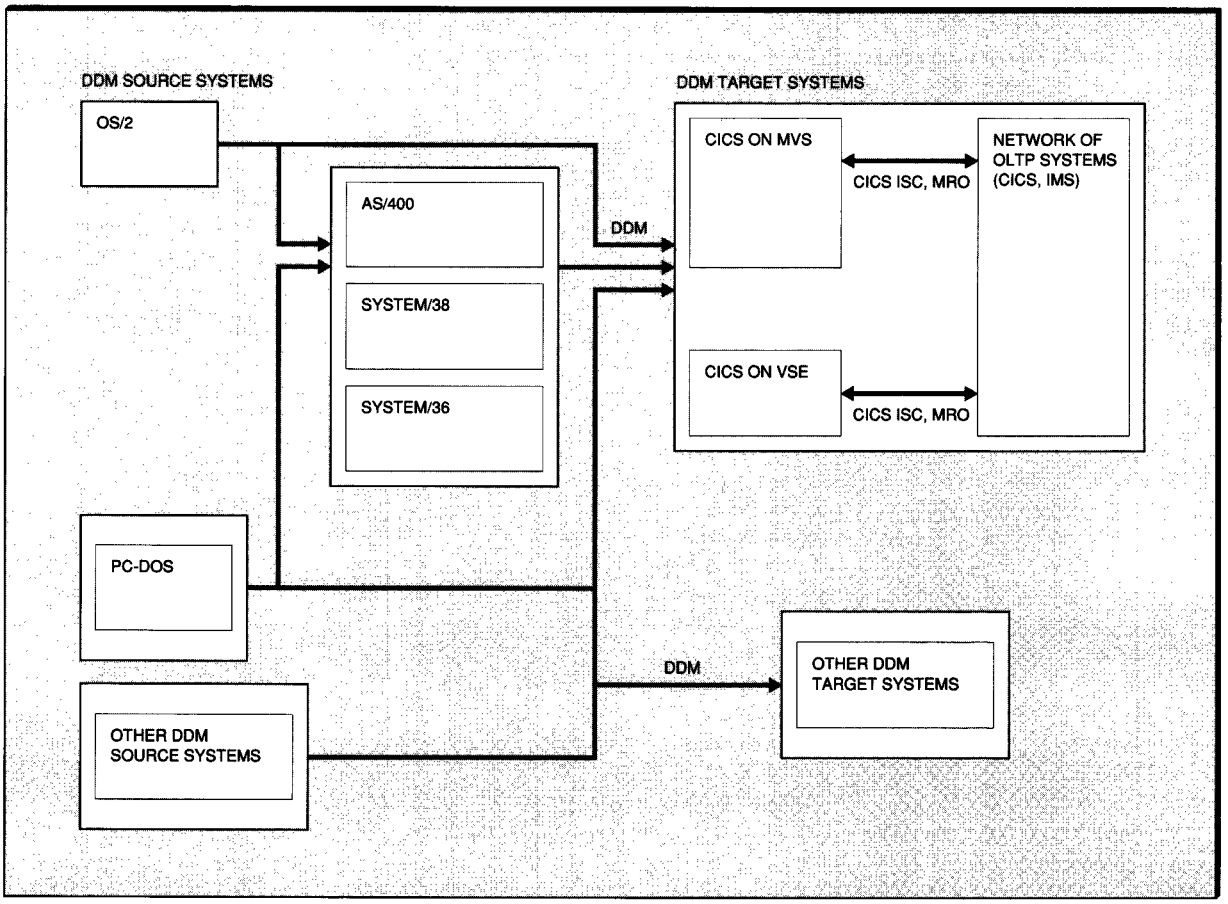
When two different systems are connected to exchange data, communication programs have to be supplied.

Because the protocol for DDM has an architecture defined, it allows new components to be added to an existing network when an enterprise grows and expands its existing configuration.

The management of data in a distributed network has been discussed in the literature for several years. It is being addressed by Open Systems Interconnection (OSI), UNIX**, Open Systems Foundation Distributed Computing Environment (OSF/DCE**) and SAA. The current focus of SAA is on the implementation of cooperative processing[6] as a subset of a generalized distributed system, including distributed files and distributed relational databases. Distributed data access helps to implement the distribution of functions in an SAA application as outlined in Reference 7.

Connecting heterogeneous file systems. Because of the variety of application programming interfaces (APIs) for file access that have evolved, DDM provides a generic file model to allow structured and independent communication between all those different file systems. This general file model allows DDM to connect heterogeneous and homogeneous file systems, while remote procedure call (UNIX, DCE) or function shipping (CICS) types of

Figure 1 DDM file access and CICS OLTP



protocols are often used to connect homogeneous file systems, i.e., systems with identical or very similar file I/O APIs (UNIX-to-UNIX, CICS-to-CICS).

DDM can connect file systems with different APIs, different functions, and even different file models, file organizations, and data representations. All file access requests and their results are translated by DDM source and target servers into a common DDM language that allows the various file systems to be connected. More details about the DDM architecture can be found in manuals and in References 1 and 2. Figure 1 illustrates how heterogeneous file systems are connected to CICS and the OLTP world via DDM implementations, whereas the OLTP systems provide their own connectivity services such as CICS multiregion operation (MRO) and intersystem communication (ISC).[8]

Data conversion. In addition, the representation of text and numeric data must sometimes be converted to allow for proper follow-on processing by the application on the source system. Applications may even require different data types or record layouts for their processing. Data conversion is of major importance when character and numeric data are shared between host-computer (EBCDIC) and workstation (ASCII) applications. Conversion of character data is part of many data management or file transfer services, whereas conversion of numeric data as a general service is almost unique to the set of DDM products. For a more detailed discussion of data conversion, see Reference 3. In today's environment, data conversion (and description) has been implemented on the programmable workstation (PWS) so that it is available to all potential target systems.

Adding new DDM components—History. The generic file model as defined in DDM allows new DDM implementations to be smoothly added to a constantly growing product set without having to modify the existing products. Availability of new DDM implementations always extended the connectivity provided through existing services and thus broadened the scope of data and systems accessible through applications on DDM source systems. In 1988, DDM was announced as the SAA CCS communication protocol to access, modify, and manage remote files in SAA. At that time, the program product CICS/DDM (MVS) was the only System/370*-based implementation of the DDM architecture. It allowed the interconnection of midrange systems (Application System/400*, System/36*, System/38*), small systems (PC-DOS with DDM/PC, NetView/PC*), and vendor software (UNIX) to System/370 hosts through CICS.

In 1989, CICS/DDM (VSE) was added as a new program product, providing access to VSE data from DDM source systems.

The availability of new SAA DDM implementations, mainly on OS/2, emphasizes the role of CICS/DDM as a service that allows SAA Common Programming Interface (CPI) applications to transparently access remote CICS data stored under MVS or VSE. This service includes the ability to copy complete CICS files or parts thereof with a single request within the systems that support DDM. Figure 1 shows the current DDM environment available to CICS/DDM. It illustrates that CICS/DDM provides a bridge from SAA systems and their users to data usually managed and controlled by OLTP systems.

Why centralized data?

Although the trend of decentralizing computer power onto a network of smaller systems brings that power closer to the end user, it is important to maintain access to existing applications and data. New business solutions are often provided on new small or medium systems. They need access to data obtained through existing applications to give added value to existing solutions and services.

The power of local workstations and departmental computers allows a variety of operations to be performed apart and independent from a central host system. Centralized systems gain more and more importance through their ability to connect all of the users of a network with all of the information processed throughout an enterprise.

Several factors that still require the support of centralized systems and centralized data, regardless of whether the data are on a System/390* or System/370 host, an Application System/400 (AS/400*), or a local area network (LAN) server (depending on the size of the organization), are:

- Availability of the same data to all users in the network
- On-line access to data that are concurrently being modified by a large number of users
- Existing applications and organizations
- Security
- Backup facilities
- Data administration

More discussion of distributed versus centralized processing tradeoffs in general can be found in Reference 7.

In the early days of software development, file I/O functions were already coded independent from the location of the data within the local system; that is, they were independent from the physical disk device and from the number and types of devices attached to the local system. Within a distributed system, applications now need to be written to be as independent as possible from the location of the data within the network.

This location independence of file access functions also facilitates migration from one host system to another: Either the data can be moved to a new system without the need to modify the applications or new applications can be written to access data on other existing systems, without requiring many new services in the existing network.

The mystery of on-line transaction processing

In contrast to its long existence and importance, on-line transaction processing remains a mystery to many people. This section explains major characteristics of the CICS OLTP system, in addition to the concepts described in Reference 9.

Single address space, multiple users. CICS allows many users to run applications concurrently in

the same address space. This capability is a major difference from interactive systems such as TSO or CMS, where users are isolated from each other by means of separate address spaces or virtual machines. It also explains one technical origin for the important role of CICS in a system with very few address spaces like VSE, which supported a maximum of only eight "partitions" until the recently announced version named Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA*). Interactive systems typically provide their users with access to private data. Sharing data between users is usually possible only on a file level, and it requires specific action to be taken by the end user.

A CICS transaction represents the active part of a program as executed by a user.

Sharing the latest information on line. OLTP allows users to share the latest information available in an enterprise. In order to implement OLTP, CICS provides a programming environment so that CICS applications can read and write records of individual files, thus minimizing the interference between concurrent users. In order to let users see the latest information as soon as possible, and to prevent one user from blocking another user, OLTP applications must hold record locks for as short a time as possible. This time is managed by CICS, which enqueues and dequeues on records implicitly when data are to be modified.

Data consistency. Operations as issued by an end user often require multiple modifications in more than one file. To prevent other users from viewing inconsistent data, such changes must be made available in atomic form to concurrent users. For example, two separate customers of a travel agency may only want to book a flight that includes a free hotel room and rental car. All these operations must be booked and confirmed at once to avoid the situation where one agent obtains the last hotel room and another one obtains the last car, with the result that neither one of the customers decides to book the travel. That is, either all modifications must be available to the user community or none must be available. This requires the concept of an LUW (logical unit of work) to be available to applications, so that either a *commit* can be performed on all updates by the application or a *rollback* can be performed in the event of a failure. CICS implements the LUW concept for all resources that may be specified as "recoverable," such as Virtual Storage Access Method (VSAM) files, databases, CICS temporary storage or transient data queues, or CICS messages. A CICS sync point command can perform either a commit or a rollback for all changes to recoverable resources. Updates to recoverable resources are logged on a journal that allows a "backout" of the changes when a failure occurs. Within an LUW, record locks on recoverable files cannot be released before the LUW ends in order to allow changes to be backed out.

These dependencies between users, applications, and data explain why CICS data are typically modified only through pretested programs. These programs are carefully introduced into production environments by control of a CICS system administrator. The same rules apply to remote applications that access CICS data, for example, via DDM, to maintain the consistency and integrity of data and to prevent unnecessary waiting on the part of concurrent users. Thus, OLTP applications build the front end to the data.

Transactions. A CICS transaction represents the active part of a program as executed by a user. To maximize transaction throughput, CICS applications are often designed as a set of small and independent "pseudoconversational" transactions that avoid prolonged usage of system resources during user "think" time. When applications communicate with each other via LU 6.2 conversations, the duration of a conversation determines the lifetime of the CICS transaction. This method leads to longer-running transactions that perform more complex operations on data. There is an obvious tradeoff between the overhead involved in allocating and deallocating conversations (including authorization of users) and the potential of saving system resources by minimizing transaction lifetime.

It is important to understand that LUWs and record locks are to be bound to individual trans-

actions in order to minimize the waiting time of concurrent users during the time when some users are thinking. Application designers and programmers have to ensure that locks are held only for the time in which a transaction is active, rather than during the possibly long terminal I/O or thinking times that may even be prolonged to coffee breaks.

Reentrancy. A CICS transaction executes as reentrant or quasi-reentrant. Multiple instances of the same program can be active concurrently when several users execute the same transaction, e.g., the booking of a flight.

Security. Typically, data are shared by a group of users and their applications. Access to CICS data from applications can be controlled by the CICS system administrator with help from CICS security features or external security managers like the Resource Access Control Facility (RACF*). The administrator can authorize trusted source system programs to access target system CICS data, thus securing the integrity of data from unauthorized access through concurrent CICS transactions and their users.

SAA distributed access to CICS data

SAA distributed data cover two types of data management systems, flat files and relational databases. CICS supports a variety of data stores, such as:

- VSAM data set, accessed through CICS file control
- Hierarchical databases (IMS on MVS, DL/I on VSE), accessed locally or through IMS database control (DBCTL)
- Relational databases (DATABASE 2*, or DB2*, and Structured Query Language/Data System, or SQL/DS*)
- Transient data queues (TD-Q) and temporary storage queues (TS-Q), both CICS-specific and either temporary or permanent data stores

Although distributed access to relational databases can be managed exclusively by the database management systems using Distributed Relational Database Architecture* (DRDA*),[10] this access is different for distributed files mainly for two reasons:

1. The CICS API adds OLTP functions to the native VSAM API through CICS file control and CICS sync pointing functions.
2. CICS has exclusive access to VSAM files and lets the data be shared with concurrent CICS users but not with other users outside the CICS address space.

Therefore, distributed files require an implementation in CICS, whereas DRDA does not, as outlined in Figure 2. CICS/DDM provides this access to CICS file data that could otherwise not be made available through direct communication between the source system file manager and VSAM.

As for the other types of data, several possible access paths can be considered for SAA CPI applications, such as:

- SQL access to hierarchical data (e.g., via data propagation)
- File access to CICS queues (TS-Q, TD-Q) and hierarchical data, for example, with help from CICS/DDM user exit code

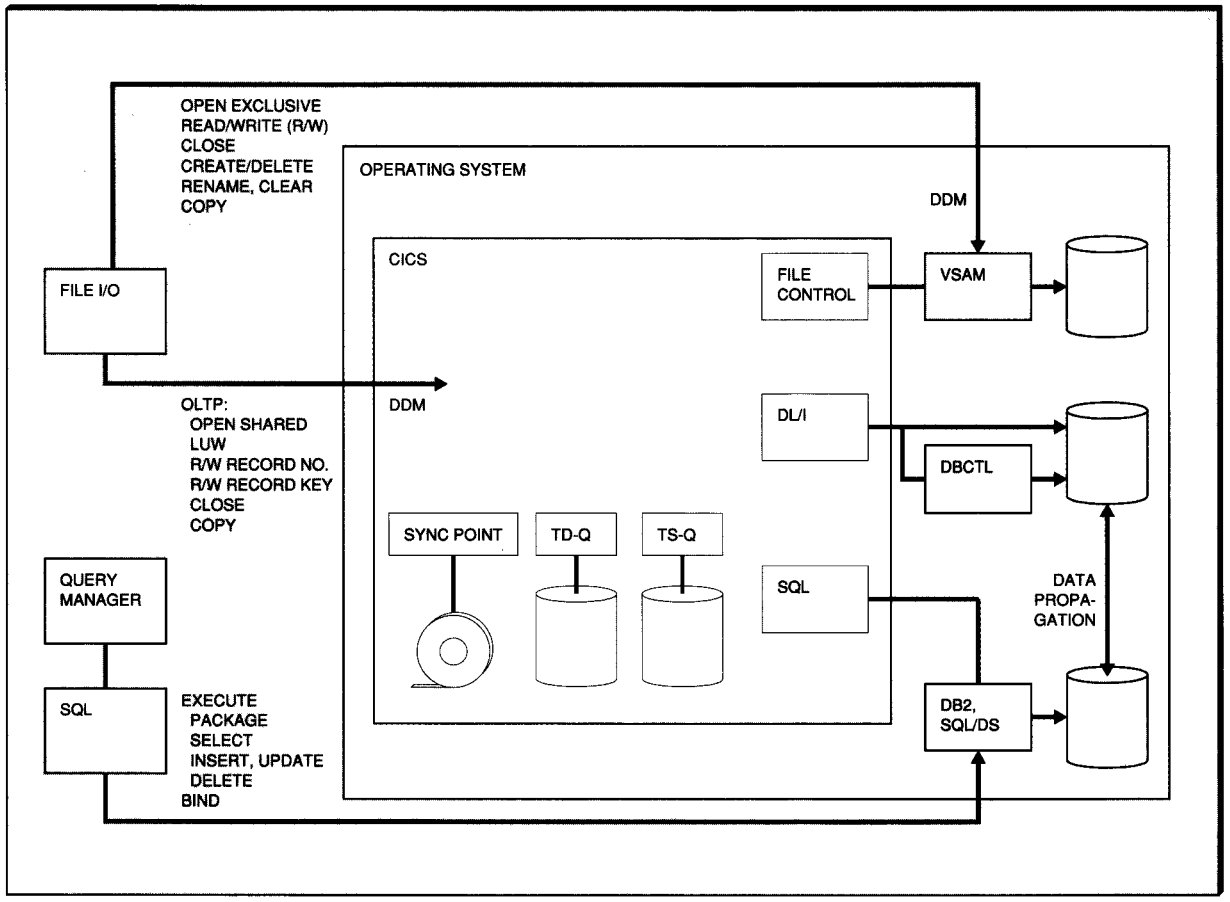
A DDM target server on CICS under MVS and VSE

CICS/DDM provides a DDM target server that allows a DDM source server to interact with CICS as it would with any system using DDM. In this way, systems that have implemented a DDM source server (see Figure 1) can access data on systems that have implemented CICS and CICS/DDM. Applications on these systems can use APIs that are supported by the DDM implementation on their system, such as the COBOL file I/O statements; there is no need to use the EXEC CICS command-level API instead.

CICS/DDM allows source systems to:

- Perform record-level access on CICS data, typically via application programs that open the file for concurrent updaters and perform read and write operations on individual or multiple records until the file is closed.
- Transfer all records of a CICS file or parts of it with a single DDM request, in either direction, often supported via operating system commands (COPYDATA on Operating System/400*, for example) on the DDM source system.

Figure 2 SAA access to CICS data



CICS/DDM is a set of CICS command-level programs, thus providing a high level of portability across the various versions and releases of CICS systems, from Version 1.7 to Version 2 up to CICS/ESA Version 3. Running as standard CICS transactions makes the product eligible for use by other CICS application services such as security, transaction routing, debugging, installation, and recovery.

CICS/DDM to support SAA file access to OLTP data. Figure 3 illustrates how DDM source systems are connected to CICS as a DDM target system by means of LU 6.2 and DDM links. It includes PC-DOS with remote DDM record file access through the DDM/PC product and OS/2 with a record API (abbreviated to RLIO/2) and a DDM source server implementation (Distributed File Manager/2, abbreviated to DFM/2). PCS is the PC Support product available for the AS/400 midrange system and its predecessors, System/36 and System/38; it may be used together with the other shown DDM source servers on PC-DOS and OS/2. Applications on a DDM source system can read and write data that could otherwise only be accessed from CICS terminals through CICS application programs. The file I/O requests of an application are translated by services on the DDM source system into DDM requests that are sent to the CICS/DDM target server. CICS/DDM retrieves the requested CICS data and sends the data back to the source system. In a network of connected CICS address spaces, a CICSPLEX, CICS/DDM is to run in the address space that contains the file to be accessed to minimize network traffic. (One DDM request may result in more than one CICS API call.)

Figure 3 DDM access to shared OLTP data

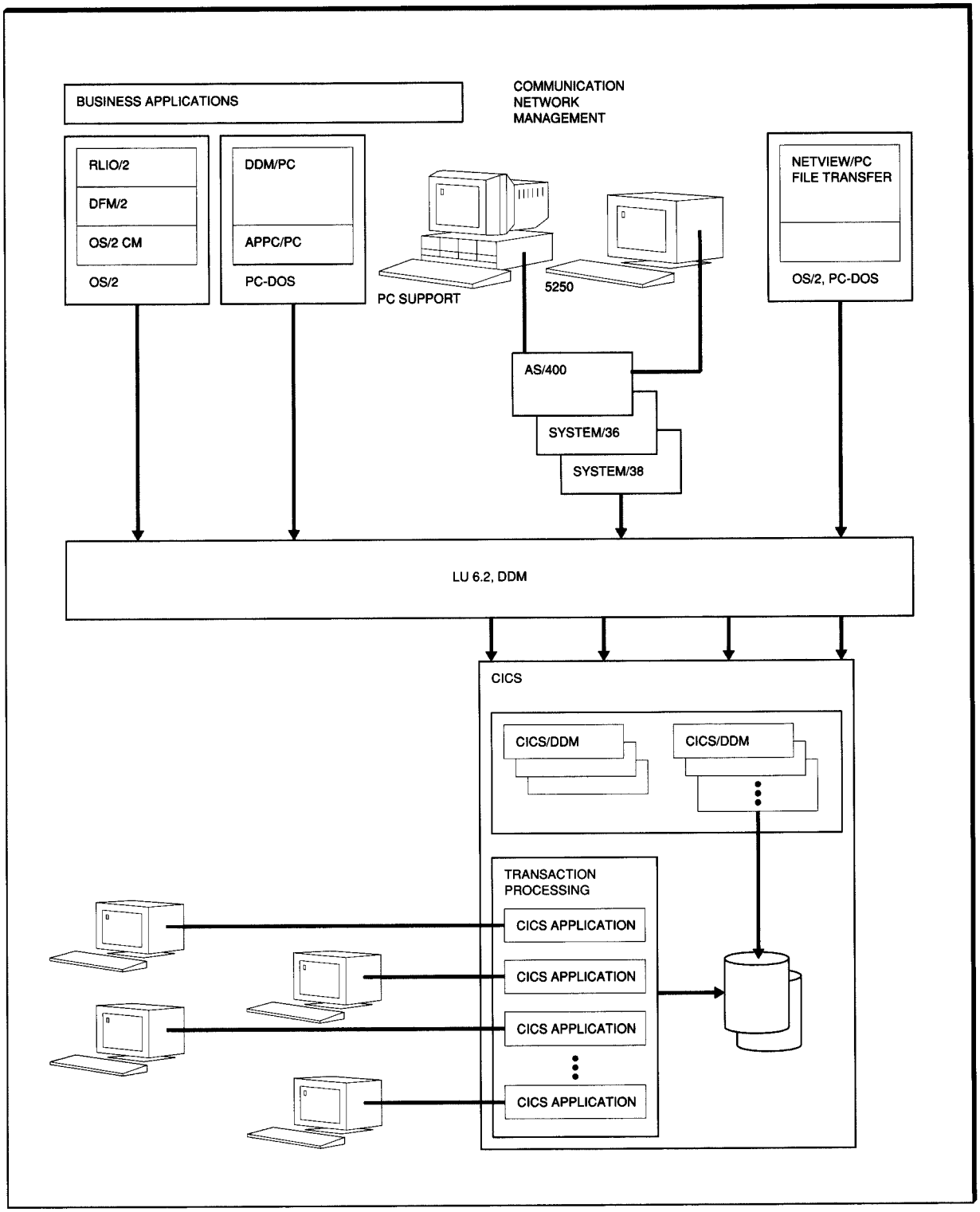
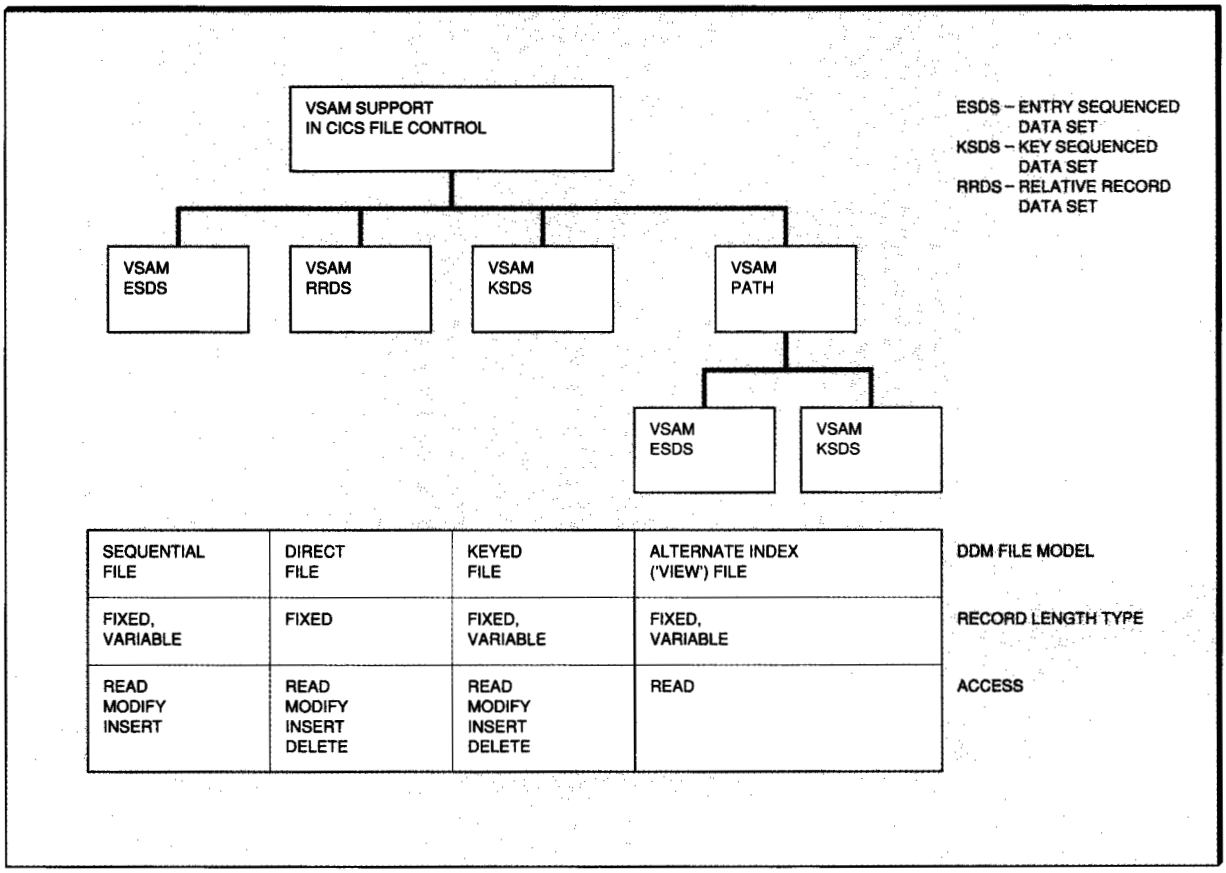


Figure 4 CICS VSAM files in DDM terminology



SEQUENTIAL FILE	DIRECT FILE	KEYED FILE	ALTERNATE INDEX ('VIEW') FILE	DDM FILE MODEL
FIXED, VARIABLE	FIXED	FIXED, VARIABLE	FIXED, VARIABLE	RECORD LENGTH TYPE
READ MODIFY INSERT	READ MODIFY INSERT DELETE	READ MODIFY INSERT DELETE	READ	ACCESS

Each active DDM source application is connected to its own instance of the CICS/DDM target server transaction, thus allowing concurrent users and jobs to access remote CICS data via DDM concurrently.

The following two subsections explain how the conceptual components of DDM are mapped to the implementation of CICS/DDM.

DDM access to CICS data. CICS supports a variety of data management systems including files, relational and hierarchical databases, and CICS-specific queues. Although the relational databases DB2 and SQL/DS have been designed to share data (even on the field level) between multiple address spaces, it is CICS that provides functions such as record-level locking and LUW for VSAM. This introduces the need for a DDM target server under CICS in order to provide on-line access to CICS data that may be shared with concurrent CICS applications.

CICS/DDM Version 1 allows all systems implementing DDM to access the following recoverable or nonrecoverable types of CICS-controlled data:

1. CICS VSAM files—CICS/DDM maps the DDM commands for sequential, direct, keyed, and alternate index files as received from the DDM source server onto the CICS file control API. Mapping allows DDM source system applications to access all types of CICS-controlled VSAM files as illustrated in Figure 4, where a CICS file refers to a VSAM data set that is defined to CICS.
2. Other files, via a file access user exit—The file access user exit allows source system applications to invoke user-written CICS application programs that access CICS data. The exit is

designed to intercept CICS API requests (including sync point and mass insert) that would otherwise be executed by CICS/DDM while accessing a CICS VSAM file. These CICS API requests are passed to the user-exit routines, which can then provide file processing or other CICS application functions independent from CICS/DDM. Existing CICS programming skill can be used to write such exit programs.

Components of CICS/DDM target server processing. The DDM target server function is comprised in a single CICS transaction. The CICS/DDM target server transaction is started when a DDM source server allocates an LU 6.2 conversation to the DDM target server in CICS. This target server is identified by a unique Systems Network Architecture transaction program name (TPN), which identifies the target server. The transaction is terminated together with the conversation either explicitly by the source server or in error situations. The DDM target server transaction receives DDM data streams, maps them on the CICS API, and lets CICS execute the request. The result of the API call is then converted into a DDM reply data stream and returned to the DDM source system. Figure 5 identifies individual DDM components of the CICS/DDM target server transaction and further illustrates how DDM requests are processed by CICS/DDM. The numbers in parentheses in the figure represent the following steps:

1. The DDM source server exchanges DDM data streams via LU 6.2 with the target system.
2. CICS invokes the DDM transaction as identified through the TPN.
3. The CICS/DDM communications manager processes the LU 6.2 conversation through the CICS terminal control API. It waits to receive requests and passes on the DDM part of the data stream for further processing. After completion by the other components, it returns the reply via LU 6.2 to the source system.
4. The CICS/DDM agent parses and builds DDM data streams and routes control to specific command processors that implement the various DDM managers and process the requests.
5. The CICS/DDM directory manager checks the existence and attributes of a file. It either queries the CICS file control table or lets a file access user exit program provide the information.
6. The CICS/DDM file access manager performs read and write operations on CICS files through the CICS file control API. It includes functions for DDM load and unload requests to transfer mass data, for example, using the CICS mass insert option.
7. To access files that were identified as a user exit file in step 5, control is passed to a user-provided CICS program.
8. Finally, CICS/DDM converts the results of the operations into DDM reply objects and reply messages, which are returned to the source system through the CICS/DDM agent and communications manager and the CICS advanced program-to-program communications support.

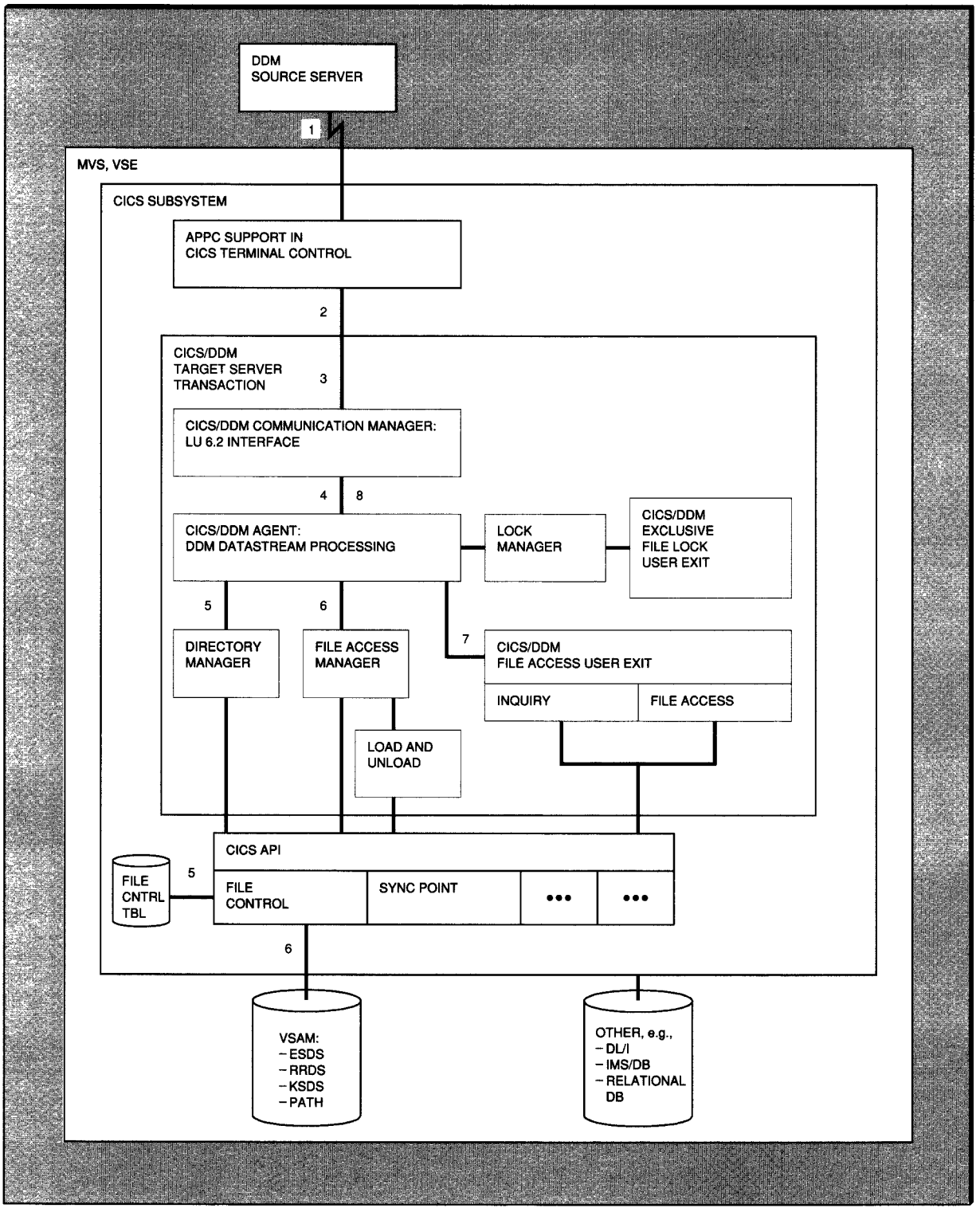
Several instances of the target server transaction can be active concurrently, where each instance uses a separate LU 6.2 conversation and represents a user, task, or job on the source system.

Duration of a CICS/DDM transaction. CICS/DDM acts as one long-running transaction, which uses all of its resources from the first remote file request until the last one has been completed. The duration is controlled by the source system and its application. In the case of a record I/O application, it may take from the first Open command on a file until the last Close command or application termination. In the case of a load or unload operation, the transaction may only be active from execution of the request until the last record has been returned.

Performance, caching, and parallel sessions. Line speed, number of communication controllers involved, and the amount of data transferred are the main factors for remote file I/O performance. The amount of CPU time consumed by CICS/DDM influences the number of source system jobs that can perform remote file I/O operations in CICS at the same time. Finally, the number of record locks held by source system applications and the time they take determine the impact on concurrent CICS or source system users of the same data. Source and target system caching techniques as provided in CICS/ESA (data tables feature) and OS/2, as well as the use of parallel sessions and multitasking features on the source system, can be used to improve the performance of the distributed file system.

Problem determination. A user-enabled trace facility is provided with CICS/DDM and can be switched on or off for all or just for named LU 6.2

Figure 5 CICS/DDM target server processing



connections to DDM source servers. All LU 6.2 conversation data received and sent together with additional information on internal control and data flows are logged on a single trace file used by all active instances of CICS/DDM. The trace data can be formatted using a CICS transaction provided by CICS/DDM, thus improving the readability for terminal displays and printouts.

The past sections reviewed relevant CICS functions and the implementation of CICS/DDM. The subsequent sections discuss the aspects that need to be considered when remote applications access CICS data. Special emphasis is first given to data integrity, followed by other considerations such as file naming, commands supported, and file models.

Ensuring data integrity from DDM source systems

On-line transaction processing introduces a need to ensure the integrity of data. Users want to see the latest information immediately, and multiple users must be able to modify different parts of a file concurrently, for example, the accounts of different customers. Changes must not be lost in the event of a failure, and related modifications must be made available to other users at the same time. To fulfill these requirements, OLTP systems support functions that are elsewhere implemented only in databases, namely the concept of a logical unit of work (LUW) and the concept of record-level locking, both of which are related to each other.

The following subsections discuss how these functions are supported by CICS/DDM, allowing SAA applications to share OLTP data.

Locking. CICS is a programming environment that allows all users to share the same data and that makes the latest updates immediately available to concurrent users. CICS does not support the concept of users being able to lock a file entirely against concurrent use between Open and Close commands. Rather, the VSAM share options can be used to lock the VSAM file exclusively for CICS to ensure full integrity by shielding the file against usage from other address spaces. All CICS applications share the CICS VSAM files with concurrent updaters, and record-level locking is used.

This specific need for OLTP impacts the source system programs that access remote CICS data, because they have to follow the same programming conventions. DDM source system applications must use record-level locking to update CICS files, and they should keep the time for record locks as short as possible. That is, source system programmers may have to be made aware of the record-level locking used by CICS and incorporate it into their design.

As an example, an attempt to lock an entire CICS VSAM file exclusively from an Operating System/400 (OS/400*) application program must fail because this function is not supported by CICS. CICS/DDM would return a DDM "File In Use" reply message (FILIUSRM), which is mapped onto a corresponding application return code. However, when there is a need to have exclusive write access to a file stored under CICS, a CICS/DDM lock user exit program can be implemented in order to ensure the integrity of the operation in the user's CICS installation.

Commitment control, LUW. When a VSAM data set is defined in the CICS file control table (FCT) as a CICS file, it can be specified to be recoverable or nonrecoverable. For recoverable files, CICS provides a logging function that supports the concept of an LUW similar to the one supported in relational databases. This CICS function allows changes to be committed on request (SYNCPT command) or when the transaction terminates normally, or to back out uncommitted changes in the event of a failure or on request (SYNCPT ROLLBACK command).

Updates to distributed recoverable files in a network must be coordinated via a two-phase-commit protocol as described in the SYNC_LEVEL(2) tower of the LU 6.2 architecture and as being addressed in SAA by CPI-RR (resource recovery).

Updates to nonrecoverable files are not committed nor is their content predictable in the event of a system or task failure.

Although the LUW concept is currently supported only in relational database (RDB) and OLTP systems and not on most DDM source systems (neither for local nor for remote data), CICS/DDM supports access to both recoverable and nonrecoverable CICS files.

To ensure the integrity of CICS data, the following considerations must be taken into account:

- Modifications to CICS recoverable files via CICS/DDM are committed as soon as possible. Usually, a sync point is performed by CICS/DDM immediately after the successful completion of the DDM write operation to a CICS recoverable file.

  In situations where the application still holds an additional update intent on a different record of the same or a different file, this sync point is deferred by CICS/DDM because CICS would otherwise release the record lock. The sync point is then performed as soon as the latest record lock is released. This phenomenon is known as the CICS/DDM "deferred sync point processing."

- Source system applications that need to modify CICS recoverable files from systems that do not (yet) support commit or rollback verbs can exploit this deferred sync point processing in CICS/DDM for their purposes. It allows them to control explicitly when sync points are performed. By means of a CICS/DDM file access user exit, a generalized service can be implemented that allows applications to easily migrate to CPI-RR or similar APIs when they become available on the source system.

Security. CICS/DDM implements the security tower of the LU 6.2 architecture in conjunction with the security and auditability features of MVS, VSE, and CICS. Therefore, the system is secured against unauthorized access as follows:

- Since CICS/DDM is a normal CICS application program, it is eligible to use all available CICS or RACF security facilities that are available to CICS LU 6.2 transactions:
  - With incoming LU 6.2 requests, access can be controlled at *bind* time (session establishment), at *attach* time (conversation establishment), or at the resource level, as described in Reference 8. Control of access prevents the attachment of unauthorized systems to CICS, controls the authority of specific users on specific systems to initiate a transaction, and restricts access to certain CICS resources.
  - A CICS security key can be defined for the DDM target server transaction to control which user or system can invoke this service.
  - By means of resource security level (RSL) checking (discontinued with CICS/ESA 3.2.1) a user can associate the appropriate RSL key with each resource that the transaction will access; such resources can be CICS VSAM files, CICS transactions, or CICS programs, for example.
- In addition, security facilities on the source system can be used to restrict access by specific source system users to specific remote files.

Access from DDM source systems

This section discusses differences in the various programming environments such as applicable commands, the meaning of the Open and Close commands, stream files, directories, and file names. It includes considerations for applications to access CICS data.

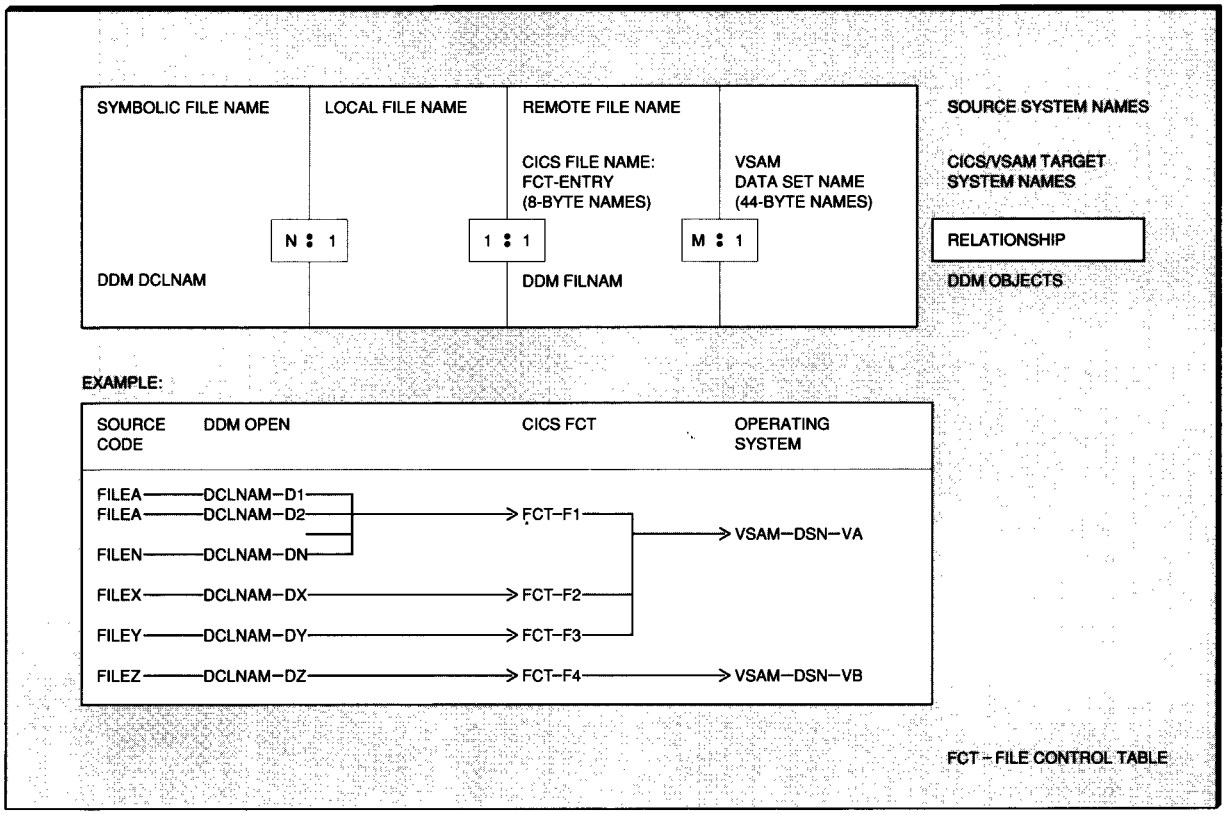
Addressing the remote CICS file. Application programs may use a variety of file name declarations to address a specific file or data set, independent of the location of the device on which the file is stored. When the application is executed on the system where the file resides, at least one symbolic name is used in addition to the physical file name:

- A symbolic name is often used in the source code of an application program to be independent from physical naming conventions.
- The name by which the file is actually known to the file system should be specified only once. Often, symbolic and physical file names are linked only at execution time of the program, thus allowing the file name to be changed without having to recompile the program.

Accessing remote files adds a new dimension to this file name mapping process, because the names must be mapped onto the file name conventions of the target system. Figure 6 illustrates this relationship between file names used in the source system application and those known to CICS.

- DDM Declare Name—The symbolic file name is mapped into a corresponding, but not identical, DDM DCLNAM (Declare Name) in order to identify and process different openings of the file on the same file name. Multiple Declare Names can be generated for a specific local file name (n:1 relationship; many-to-one). The Declare

Figure 6 Multiple access paths to the same CICS file



Name is not seen by the user, but it is part of the DDM data stream.

- DDM File Name—The remote file name is either directly specified by the application or mapped from the local file name by the DDM source server or a related service. In both cases, the relationship between local and remote file name is one-to-one. The remote file name is processed as a DDM FILNAM (File Name) and is used to identify the file within the target system. To access remote CICS data, FILNAM must be a valid CICS file name.
- CICS file names and VSAM data sets—In CICS, the file name must be defined in the CICS FCT, where multiple file names may again be specified for the same physical VSAM data set. This additional indirection (m:1 relationship; many-to-one) allows control of the access characteristics of different CICS applications to the same VSAM data set, e.g., the operations to be performed by the application.

Open and Close. CICS applications do not need to explicitly open or close files, because CICS itself opens and closes VSAM data sets for access through CICS applications that have been authorized to perform the requested operations. This operation is different from when SAA file I/O applications access remote CICS data via DDM. The Open request identifies the file as a remote file and specifies the access intent of the application to the data and the degree of sharing with concurrent users. The access intent can be used to check whether the CICS file name has been defined to allow the requested operations to be performed (SERVREQ entry in CICS FCT). The sharing parameter must be set to allow concurrent updaters. The Close request terminates the access of the application to the remote CICS file name.

Managing CICS files (create, delete, rename, clear). As CICS files are usually not owned by a specific user but instead are used to share data on

line, no CICS API functions perform file management operations such as create, delete, rename, or clear on CICS VSAM files. Rather, a CICS system administrator creates and deletes VSAM data sets and provides the necessary CICS definitions to make them available as CICS files. Consequently, such operations are not supported by CICS/DDM for execution from a DDM source system. The only file management operations supported are load and unload, thus allowing copy operations to be performed on CICS data.

The introduction of PWS into the network raises a requirement to exploit standard PWS applications and services.

Cooperative processing

Together with DDM source server implementations on the PWS, CICS/DDM provides a cooperative processing[6] function as do other applications such as CICS OS/2 with a CICS host. DDM-based applications have the advantage of connectivity beyond the OLTP world, i.e., they can easily be written to access CICS data as well as OS/400 data, for example. When programmable workstations are introduced into the network, a requirement to exploit standard PWS applications and services is often raised, including support for byte stream file I/O and directory commands. The following subsections discuss how these functions apply to CICS data and OTLP and how PWS applications can access the contents of CICS files with support from their local DDM component.

Directories. Programmable workstations usually implement the concept of hierarchical directories with one root on each drive. They allow the files to be grouped on the system to correspond to both the logical relationships of files and the location of files.

- Grouping files according to their logical relationships is as important for users who need to change the contents of files interactively as it is for system personnel to control the inventory of files and programs on a system.
- The location of a file on physical I/O media is always very important information, for security reasons as well as for addressability. When remote resources are introduced into the network, it is usually the drive letter that is assigned to a remote system.

In OLTP environments, the end user typically invokes a program that in turn accesses and modifies the data. Thus, support for a "dir" command that lists all or a subset of the files in a remote CICS system is not so important as in interactive environments and is currently not supported by CICS/DDM.

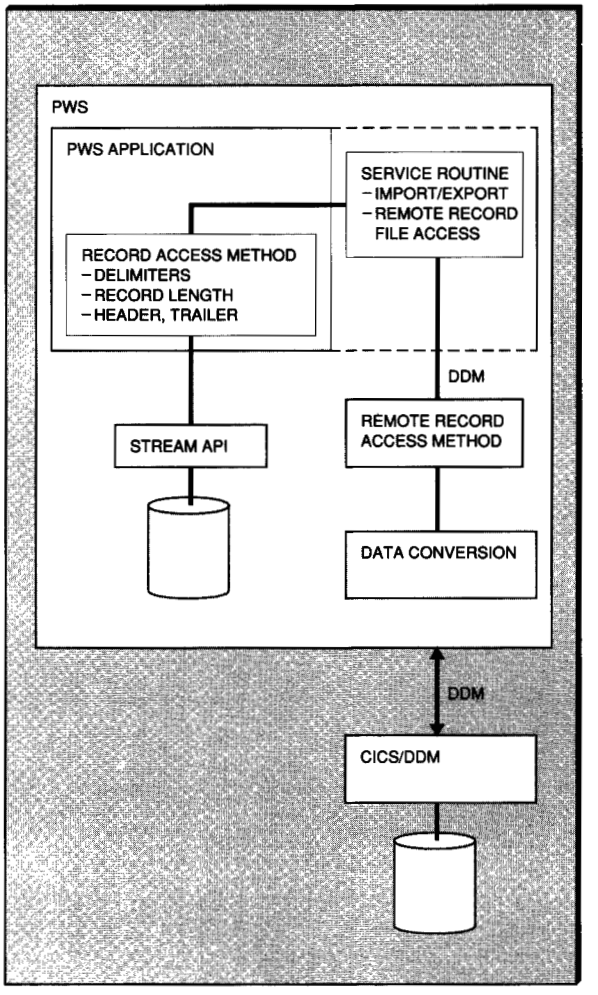
Stream file access. Discussions about the purpose and advantages of byte stream files, record files, and databases can be found in References 11, 12, 13, and 14. The representation of files as linear byte streams gives programmers considerable power and flexibility to choose whatever structure is best-suited to a particular application. This freedom of choice, however, often results in application-specific access methods being developed for programs that require random access to defined substructures within a file, such as records and fields, thereby increasing application size, complexity, and development time. The same arguments apply to databases, which add an additional support level to applications, compared to record files. They often provide improved data sharing (on the field level), data integrity and consistency, and help to minimize data redundancy, but often at the cost of performance (see Reference 14). This is one reason why record files are sometimes still preferred compared to databases, in addition to the amount of data already available on files.

Connecting PWS stream file systems with host record file systems adds the question of how standard PWS applications can operate on host data. Simple existing solutions providing a file copy facility like IND$FILE apply only to interactive systems such as CMS and TSO with their corresponding file systems. They are not appropriate for keyed files nor for data that are shared between concurrent writers. In the following, we explain how PWS applications can solve this problem when accessing the OLTP environment provided by CICS.

- Application-specific record access method—PWS applications written for stream file access often provide their own private record access method by implementing application-specific conventions (delimiters, header and trailer records, fixed record lengths, etc.). This application internal record file interface is to be used to import and access data from other sources that require record access methods, such as the data managed in an OLTP system. (See Figure 7.)
- Write access to shared data—When a PWS application is to modify OLTP data shared with concurrent writers of the same file, a record access method is to be used in order to minimize the interference between concurrent users. In this case, the PWS application has to follow the same rules that apply to OLTP applications in general. This type of data access is the typical scenario for business applications that require an OLTP environment. (See Figure 8.)
- Write access to private or temporary storage—Sometimes an application has to collect input for further processing by other CICS applications that are written to manage the access to shared files. In this case, information has to be provided in an intermediate store between two applications that communicate with each other. This is a typical scenario for which CICS temporary storage queues have been designed. With CICS/DDM, a PWS application may access a CICS temporary storage queue as a sequential file. The NetView/PC bulk data transfer facility is an example of this scenario. (See Figure 9.)
- Read access—Often, a PWS is used to provide business graphics or spreadsheet calculations. Such applications only require part or all of a host file to be extracted, thus providing a snapshot of OLTP data. To import the contents of a CICS file that is in record format, the PWS application has to import the requested data and convert the records into the byte stream format of the application. This processing depends either on selective read operations on individual records that form the extracted file or on a full file read operation that goes up to at least the latest record accessed. (See Figure 10.)

To integrate the functions of PWS applications into an OLTP environment, service routines are required that import the contents of a CICS record file into the byte stream file format of an application. Such routines allow remote CICS data to be extracted for follow-on processing through PWS applications, thus providing a snapshot of host data.

Figure 7 Stream application-specific record access



Providing PWS applications as a front end for CICS OLTP requires a record access method as the interface to CICS applications and their users. In one case the PWS application shares CICS files on line with concurrent CICS users. In this situation it needs to access individual records and has to minimize the time during which record locks are held. In another case, the OS/2 application provides only input for further processing by other CICS applications that control the access to the shared files. In this latter case, information has to be provided in an intermediate store between two applications that communicate with each other.

Figure 8 Sharing data between concurrent writers of the same file

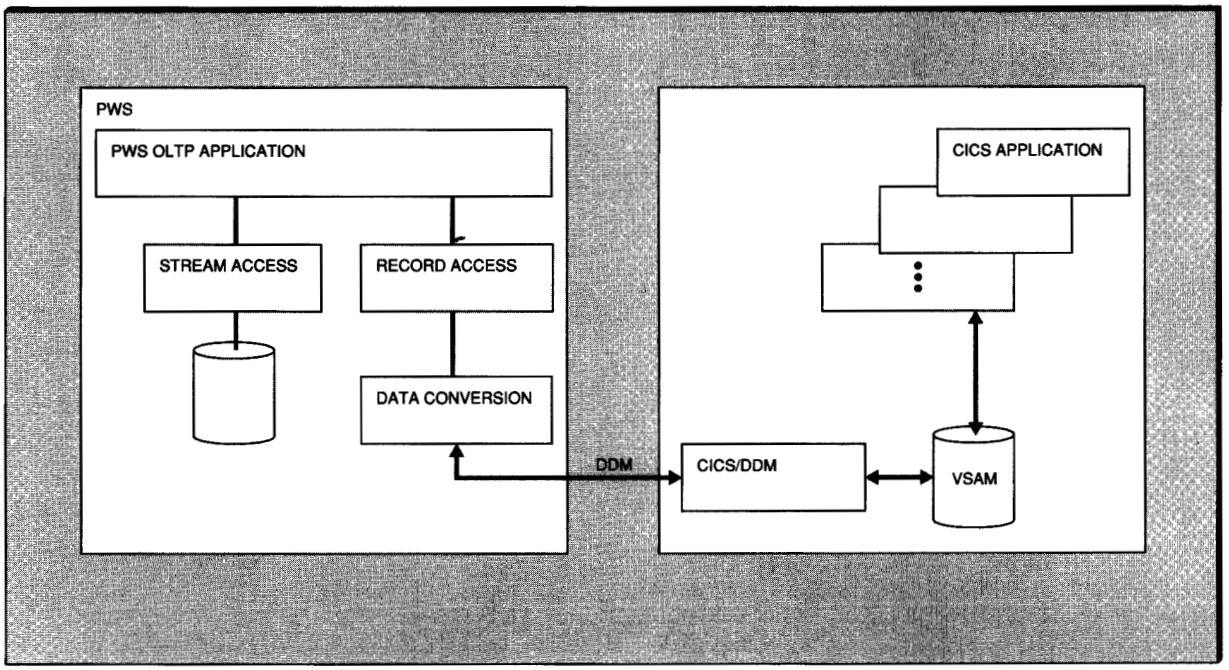


Figure 9 Providing input for a CICS application

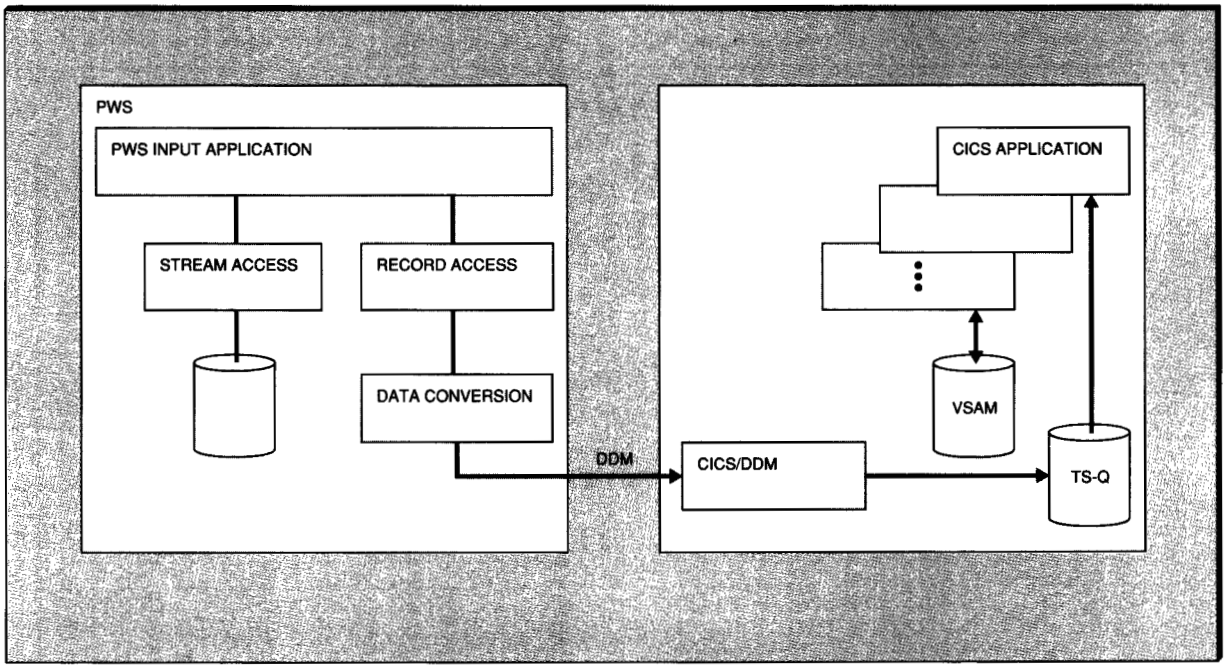
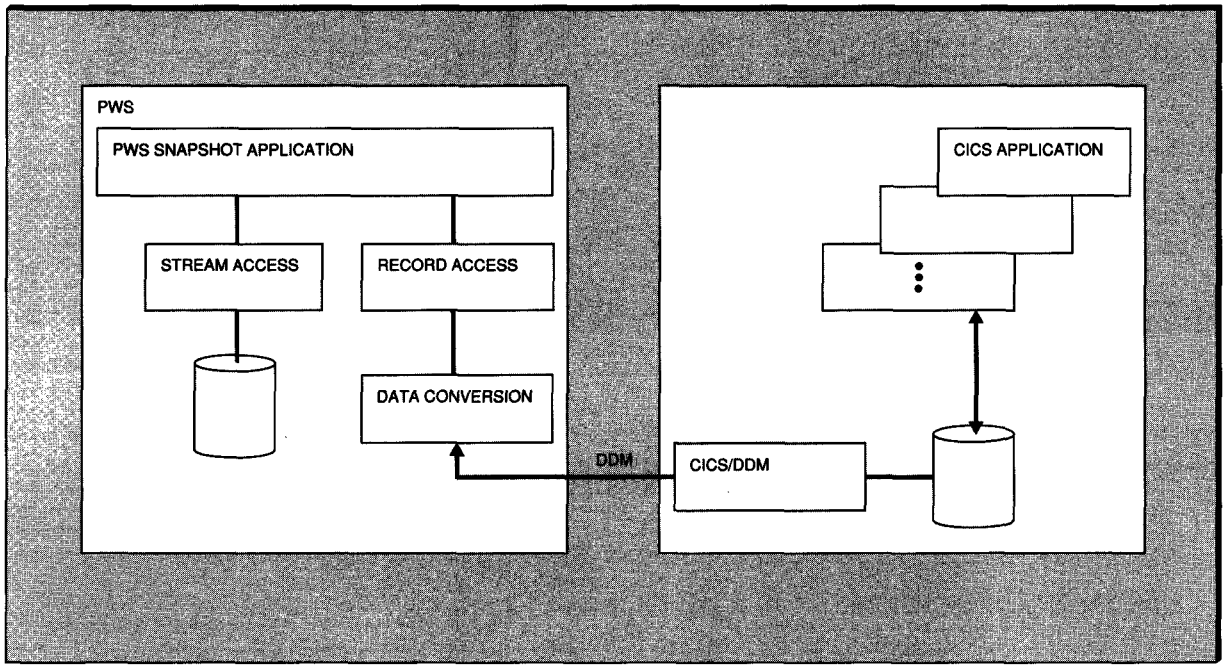


Figure 10 Extracting CICS data for processing by PWS applications



CICS temporary storage queues have been designed for such a typical scenario. In both cases, the data passed to CICS must be in record format to allow further processing by CICS applications. Information provided by standard OS/2 applications in byte stream format must be extracted and mapped into the more regular structure of a record file.

Cooperative or distributed applications designed to connect OS/2 with SAA environments may use CICS/DDM to share information with CICS applications in the same way as they do with users and applications on systems such as OS/400 and MVS. NetView/PC is an application that uses CICS/DDM as a data transfer vehicle to provide input for further processing by related CICS applications; for example, call detail information collected from various devices such as computerized branch exchanges and private branch exchanges can be forwarded to a CICS host for billing and for traffic line optimization programs (see References 15 and 16).

Concluding remarks

The specific characteristics of an OLTP environment require source system applications to use a record access method. To minimize interference between concurrent modifiers of the same data, the concepts of record locking and LUW have to be applied in these programs. These concepts ensure data consistency while maximizing transaction throughput and data availability.

PWS applications can access the contents of CICS files via CICS/DDM and convert the contents to (or from) the byte stream format of the application with help from their local DDM component.

We have shown that CICS/DDM connects heterogeneous file systems to the data managed through the OLTP environment provided by CICS. In SAA, it provides CPI and system applications with location-transparent access to CICS data. Applications can access CICS data (on MVS and VSE) with no or minimal change compared to local data access and access to remote data on any other environment that has implemented DDM.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of UNIX Systems Laboratories, Inc. or Open Software Foundation.

Cited references

1. R. A. Demers, "Distributed Files for SAA," *IBM Systems Journal* **27**, No. 3, 348–361 (1988).
2. R. A. Demers, J. D. Fisher, S. S. Gaitonde, and R. R. Sanders, "Inside IBM's Distributed Data Management Architecture," *IBM Systems Journal* **31**, No. 3, 459–487 (1992, this issue).
3. R. A. Demers and K. Yamaguchi, "Data Description and Conversion Architecture," *IBM Systems Journal* **31**, No. 3, 488–515 (1992, this issue).
4. V. Ahuja, "Common Communications Support in Systems Application Architecture," *IBM Systems Journal* **27**, No. 3, 264–280 (1988).
5. L. A. Buchwald, R. W. Davison, and W. P. Stevens, "Integrating Applications with SAA," *IBM Systems Journal* **27**, No. 3, 315–324 (1988).
6. E. F. Wheeler and A. G. Ganek, "Introduction to Systems Application Architecture," *IBM Systems Journal* **27**, No. 3, 250–263 (1988).
7. A. L. Scherr, "SAA Distributed Processing," *IBM Systems Journal* **27**, No. 3, 370–383 (1988).
8. *CICS Intercommunication Facilities Guide*, SC33-0133-2, IBM Corporation; available through IBM branch offices.
9. *Transaction Processing: Concepts and Products*, GC33-0754, IBM Corporation (1990); available through IBM branch offices.
10. R. Reinsch, "Distributed Database for SAA," *IBM Systems Journal* **27**, No. 3, 362–369 (1988).
11. R. Q. Cordell II, M. Misra, and R. F. Wolfe, "Advanced Interactive Executive Program Development Environment," *IBM Systems Journal* **26**, No. 4, 361–382 (1987).
12. D. M. Ritchie, "A Retrospective," *Bell System Technical Journal* **57**, No. 6, Part 2, 1947–1970 (1978).
13. J. M. Bissell, "Extended File Management for AIX," *RT Personal Computer Technology*, SA23-1057, IBM Corporation (1986), pp. 114–118; available through IBM branch offices.
14. G. Sharman, "CICS in the 1990s—The Evolution of Transaction Processing Systems," to be published.
15. *CICS/VS and Distributed Data Management,* GH24-3157-00, IBM Corporation; available through IBM branch offices.
16. M. Ahmadi, J. H. Chou, and G. Gafka, "NetView/PC," *IBM Systems Journal* **27**, No. 1, 32–44 (1988).

General references

C. C. Barnes, A. Coleman, J. M. Showalter, and M. L. Walker, "VM/ESA Support for Coordinated Recovery of Files," *IBM Systems Journal* **30**, No. 1, 107–125 (1991).

Concepts of Distributed Data, SC26-4417, IBM Corporation (1988); available through IBM branch offices.

W. T. Fischofer, "VM/ESA: A Single System for Centralized and Distributed Computing," *IBM Systems Journal* **30**, No. 1, 4–13 (1991).

D. J. Haderle and R. D. Jackson, "IBM Database/2 Overview," *IBM Systems Journal* **23**, No. 2, 112–125 (1984).

D. L. Schleicher and R. L. Taylor, "System Overview of the Application System/400," *IBM Systems Journal* **28**, No. 3, 360–375 (1989).

R. L. Stone, T. S. Nettleship, and J. Curtiss, "VM/ESA CMS Shared File System," *IBM Systems Journal* **30**, No. 1, 52–71 (1991).

R. C. Summers, "Local-Area Distributed Systems," *IBM Systems Journal* **28**, No. 2, 227–240 (1989).

Accepted for publication March 12, 1992.

Klaus Deinhart *IBM Deutschland GmbH, GADL Boeblingen, 7030-91, Pascalstrasse 100, FRG-7000 Stuttgart 80, Germany.*

Mr. Deinhart received an M.S. in computer science at the Technical University, Darmstadt, Germany, where his main interests were in databases, theory, and open systems. He joined IBM in 1984 at the Program Product Development Center in Sindelfingen and worked on the development of several program products, during which time he held various positions. His first assignments were on videotex communication monitors that included IBM's first Open Systems Interconnection products. These systems were based on IBM large systems; follow-on projects were on System/36 and AS/400. After being involved with a project on cooperative processing between personal workstations and the Customer Information Control System, Mr. Deinhart became the team leader of a CICS/DDM development project and was a member of the DDM architecture control board. Following that work, he held technical planning and, later, business planning assignments in the Application Session Manager and Network Security area related to NetView Access Services. Currently, he is a planner in the German Application Development Laboratory Boeblingen for networking products.

Reprint Order No. G321-5484.