Technical note

IMS celebrates thirty years as an IBM product

by K. R. Blackman

This technical note recognizes the thirtieth anniversary of the IBM Information Management System (IMS™) and describes the initial requirements that led to the development of IMS. It also describes how IMS has evolved over the past thirty years to implement new technology while preserving customers' application programming investments.

In the early 1960s, President John F. Kennedy envisioned humans walking on the moon and set the goal for this vision to be accomplished before the end of the decade. Achieving this goal presented a challenge to American industry. North American Rockwell won the bid to launch the first spacecraft to the moon, and in 1965 they established a partnership with IBM to fulfill the requirement for an automated system to manage large bills of material for the construction of the spacecraft.

In 1966, 12 members of the IBM team working with 10 members from North American Rockwell and 3 members from Caterpillar Tractor started the design and development of the Information Control System (ICS) and Data Language/I (DL/I). During the design and development process, the IBM team was moved to Los Angeles and increased to 21 members. This team completed and shipped the first release of ICS.

In April 1968, ICS was installed. The first "IMS READY" message was displayed on an IBM 2740 terminal at the Rockwell Space Division in Downey, California, on August 14, 1968. ICS was renamed Information Management System/360 in 1969. Thus, IBM's Information Management System (IMS*) came into being, and the database management system revolution began.

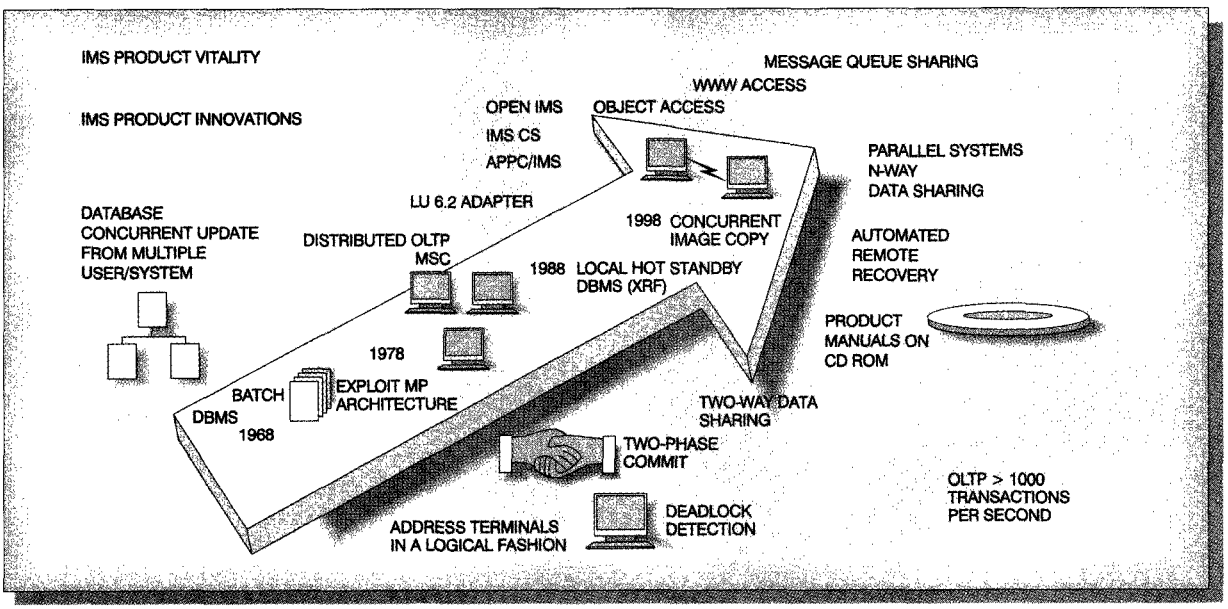
Database management system. The IMS database management system (DBMS) implemented the hierarchical model tree structure to organize the collection of records in a one-to-many entity-relationship data model. The use of a database introduced the concept of separating application code from the data. The database management system controlled the access and recovery of data. This established a new data processing technology paradigm. Now the application code could focus on the access and manipulation of the data. DL/I was developed to provide the application code with a standard interface to the data.

Data Language/I. Navigational access to the data for application code is through the DL/I standard callable interface data manipulation language (DML). Since the data control was moved from the application, an opportunity was provided for on-line access to the data.

An on-line component initially called Information Control System/Data Language/I (ICS/DL/I) was developed to support data communication access. To enable data communication transparency to the application program, the DL/I callable interface was ex-

0018-8670/98/$5.00 © 1998 IBM

Figure 1 IMS technology evolution



tended to the on-line component. To maintain the integrity of data communication messages and to provide a queuing concept for scheduling application programs, a message queue function was created. ICS/DL/I evolved to become the transaction manager function of IMS.

IMS has evolved to a family of product solutions that enables managed access to corporate business data. In 1998, IMS is celebrating its thirtieth year as an IBM software product.

IMS technological innovations. Today, over 90 percent of the top worldwide companies in the areas of manufacturing, finance, banking, retailing, aerospace, communications, government, insurance, high technology, and health care use IMS to run their daily operations.[1] Figure 1 shows the evolution of IMS and the various technologies that have been implemented over the past 30 years.

These technologies have been instrumental in enabling corporations to refine their information technology solutions to be cost-effective. This technical note describes these various technologies and how IMS has evolved over the past 30 years to provide the functionality that supports these technological innovations.

IMS database

The IMS database (DB) function provides a full-function resource manager and a fast path resource manager for hierarchical database management. These resource managers map the DL/I application call interface to the underlying operating system access methods. This mapping enables the application code to function independently of operating system access methods and to remain transparent to the physical storage of the data. The data managed by IMS are organized in hierarchical database records. A database record is composed of segments, a segment being the smallest piece of information that IMS can store. A segment contains fields that are the smallest pieces of data an application program can manipulate. A field is identified as a unique key field that can be used to navigate the database to find a specific segment. The hierarchical structure of the segments establishes the structure of the database record. A root segment identifies a database record, and a database record cannot exist without a root segment. Dependent segments are the pieces of data that complete the rest of a database record. The IMS DB full-function resource manager provides sequential access, indexed sequential access, and direct access for database processing. The fast path DB resource manager provides the direct method for

processing data by using direct access pointers to the segments.

Data integrity

One of the fundamental concepts supported by IMS DB is the concurrent update capability from multiple application programs. IMS DB contains a program isolation local lock manager to support concurrent application program processing. The local lock manager controls access to the database records to ensure data integrity by preventing duplicate access to the same record until the application program completes its processing of the record. The database record is locked when position is first obtained on the record. The item locked is the root segment. The database record is locked until position is changed to a different database record or until the application program reaches a commit point.

Because data are always accessed hierarchically, when a lock on a root segment is obtained, IMS determines whether any application programs hold locks on dependent segments. If there are no locks on dependent segments, it is not necessary to lock dependent segments when they are accessed. When an application program updates a segment, the segment, not the database record, is locked. IMS DB uses the Internal Resource Lock Manager (IRLM) global lock manager to support the global sharing of databases across systems in a sysplex. When the IRLM is used, no lock is obtained when a dependent segment is updated. Instead, the root lock is held at single update level when exiting the database record. Therefore, no additional locks are required if a dependent segment is inserted, deleted, or replaced. This established the foundation for two-way data sharing and Nways* data sharing in a System/390* (S/390*) Parallel Sysplex* cluster.[2]

Another key aspect of data integrity is recovery. The IMS logger function provides recovery and restart capability for the system. IMS logs the following events:

- When IMS starts up and shuts down
- When application programs start and terminate
- Changes made to the database
- Communication messages received and responses sent
- Application program checkpoints
- System checkpoints

The original IMS logger wrote the log records to a tape drive. The IMS logger was converted to a direct access storage device (DASD) logger to support high-volume traffic. IMS first writes log records to a high-speed DASD data set called the Write-Ahead Data Set (WADS). When IMS fills a DASD track on the WADS, IMS copies the entire track to another DASD data set called the Online Log Data Set (OLDS). When IMS is close to filling the last available OLDS, it archives the full ones to System Log Data Sets (SLDSs). While the Log Archive utility is creating an SLDS, it can also create a Recovery Log Data Set (RLDS). The RLDS contains only the log records needed for database recovery.[3] The Database Recovery Control (DBRC) facility is used to support log management, database recovery control, and data sharing.

DBRC automatically records all log data sets produced by the on-line IMS subsystem and by log archiving jobs in a pair of Key-Sequenced Data Sets (KSDS) called RECON (recovery control). IMS uses this information for database recovery jobs if databases are registered and also for IMS restart. DBRC also tracks the archiving requirements of the on-line data set (OLDS) and, if requested, generates and submits the Job Control Language (JCL) for archiving jobs.

DBRC also records:

- Database image copies
- Database reorganizations
- Database recoveries
- Accumulations of database changes
- Backouts of database changes

Because DBRC records this information in the RECON, DBRC can generate JCL for executing a database recovery.

Data sharing requires that the databases be registered with DBRC. DBRC checks on whether subsystems have authority to perform the requested task and whether other subsystems are not currently reserving the database. Concurrent access to databases by systems in one or more processors is controlled with a common (shared) DBRC RECON data set. To maintain data integrity, status indicators in the RECON data set control concurrent access and recovery actions for the databases. This common RECON data set is required in a data-sharing environment so that a given database is identified to all the sharing subsystems.[4]

IMS architectural structure and flow

IMS has evolved with the System/360*, System/370*, and System/390 architectures while preserving the ability of the application program to process data. Some customers have application programs written 30 years ago that still run today without any changes to their existing code. The fundamental architecture of IMS consists of a control region, a DLI secondary address space (DLISAS), a DBRC address space, an IRLM address space, and one or more dependent regions.

The control region is the execution environment for the IMS system software, the control blocks, and storage pools required for managing communication access and application program scheduling. The control region also contains the fast path system software for managing access to fast path databases. This isolates the IMS system functions from the customer's application programs to maintain the integrity of the IMS system. The DLISAS execution environment contains the IMS DB full-function system software, control blocks, and storage pools for managing access to the full-function databases.

The dependent regions provide the execution environments for the application programs to process transactions. IMS schedules a customer's application programs in the dependent regions to provide for parallel processing and a separation of application program processing. The scheduling of an application program in its own dependent region also provides application program integrity. When a transaction message arrives in the IMS control region, the IMS scheduling function checks for a dependent region that is available for running the application program. The available dependent region is supplied with the name of the application program to be loaded and executed in the dependent region. The application program uses the DL/I calls for message queue processing to retrieve the message from the message queue and to process the request.[5] When the application program completes the processing of the request, this indicates the completion of the unit-of-work, and the IMS sync-point manager initiates the two-phase commit process. During sync-point processing, IMS creates a log record to establish commitment of database changes and availability of output messages. The commit process is not complete until IMS physically writes this log record to the OLDS.

For Phase 1, the IMS resource managers are asked to commit updates, and for Phase 2 the IMS resource managers are told to commit or abort the changes. After the IMS sync-point manager completes the two-phase commit process, the IMS transaction manager (TM) will check on whether there are any more work requests for the application program. If no additional work is ready to be processed, IMS TM determines whether the region can become pseudo wait-for-input (pseudo WFI). If the region is eligible for pseudo WFI, the region remains scheduled for the transaction and waits until another message is entered for the region. If the next message is for the scheduled transaction, the message is passed to the application program. If the next message is for a different transaction, IMS TM terminates the application program and schedules a new application program to process the new message. This self-tuning technique is used to maximize the use of a dependent region by an application program.[6] The ability to support multiple dependent regions facilitates workload balancing and application program isolation. IMS distributes the workload among processors by off-loading some of the IMS control region task control block (TCB) work to dependent region TCBs.

The following functions operate in dependent region TCBs using the cross-memory facility of the Multiple Virtual Storage (MVS*) operating system to communicate with the IMS control region for:

- Application program scheduling
- Application program DL/I calls
- Application program sync-point processing
- Application program termination

This enables customers to utilize their CPU investment with their application processing. The IMS multiaddress space architecture exploits the multiprocessing architecture of MVS and the transformation of the S/390 platform to the S/390 Parallel Enterprise Server*. The IMS control region also provides the functions that support communication access.

Communication access

A key concept supported by the IMS transaction manager is to view communication devices and end users in a logical fashion. This concept provides transparency for the application program and security for end-user messages. The application program does not have to know the characteristics of the communication device when sending data to or receiving data from the device. The IMS TM function uses the

message queue resource manager and expedited message handler (EMH) resource manager to manage the communication messages to be processed by the application program. These resource managers support the DL/I communication message application call interface to provide a communication message to the application. The IMS TM resource managers use the concept of a message segment to manage a communication message, and multiple message segments can be used to support a complete communication message. The application programs use the DL/I interface to receive and send message segments. Another key function of message segments is to provide network transparency to the application programs. The IMS TM resource managers use the IMS TM communication functions to interface to the network protocols for communication message processing.

The IMS TM communication manager function supports the S/390 Virtual Telecommunications Access Method (VTAM*). The communication manager is the interface between the message queue and EMH resource managers and VTAM. IMS TM also provides a multiple systems coupling (MSC) feature. MSC is a private IMS TM protocol that only permits the coupling of IMS systems to other IMS systems. MSC is an extension of the IMS TM communication and scheduling capabilities and enables the user to perceive one virtual IMS system while the transactions are being routed across a complex of IMS subsystems. The MSC function can be used to provide workload distribution across a single system boundary or across geographical boundaries.

IMS TM can also be used to access other types of database subsystems using the external subsystem attach facility (ESAF). ESAF gives application programs running in IMS dependent regions the capability to obtain data from external subsystems, such as DATABASE 2* (DB2*).

Opening IMS access

In September 1990, IBM introduced the MVS systems complex, or sysplex. The base sysplex provided the cross-system coupling facility (XCF) component of Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA*). XCF services allow authorized applications on one system to communicate with applications on the same system or on other systems.

Also during this time frame, MVS and IMS were working on the requirement to enable cooperative application processing using the IBM Systems Network Architecture Logical Unit 6.2 (SNA LU 6.2) Advanced Program-to-Program Communications (APPC) protocol. IMS TM was changed to support the APPC protocols and still preserve the existing application program interface. IMS TM utilizes the APPC/MVS facilities to enable APPC support. APPC/MVS and APPC/IMS are one of the authorized application pairs to implement the XCF protocols for communication.

The IMS TM resource managers use the concept of a message segment to manage a communication message.

APPC/MVS uses APPC/VTAM macros to communicate with the communications network and uses XCF to communicate with APPC/IMS. An IMS application program can also use APPC/MVS services for direct communication processing. The coordination of processing between two APPC programs is the synchronization level (sync_level). APPC sync_level defines the protocol that is used when changing conversation states. APPC and IMS support the following sync_level values:

- NONE—Specifies that the programs do not perform synchronization
- CONFIRM—Specifies that the programs can perform confirmation processing on the conversation
- SYNCPT—Specifies that the programs participate in coordinated commit processing

IMS requires the Operating System/390* (OS/390*) Resource Recovery Services (RRS) as the syncpoint manager (SPM) when a conversation with sync_level=SYNCPT is established. RRS controls the commitment of protected resources by coordinating the commit or backout request with the participating owners of the updated resources, the resource managers. IMS is the resource manager for DL/I, fast path data, and the IMS message queues. The application program decides whether the data are to be committed or aborted and communicates this decision to the SPM. The SPM then coordinates the actions in support of this decision among the resource managers. IMS Version 6 interfaces to RRS to pro-

vide for coordinated commit. IMS assumes the role of a participant in the two-phase commit process with the sync-point manager, RRS.

Since XCF services were introduced to provide a standard MVS sysplex communication protocol, IMS implemented the Open Transaction Manager Access (OTMA) function. The OTMA function uses XCF application programming interfaces to define a transaction pipe (TPIPE) connection between an OTMA client and IMS TM OTMA server function. The TPIPE supports a full-duplex protocol. An OTMA client is an MVS application program that sends transactions to an IMS server and receives the output. The MVS application program must be a member of an XCF group that includes IMS and uses the OTMA protocol.[7] OTMA also works with RRS to support coordinated commit processing. In an OTMA environment, OTMA is not a resource manager registered with RRS. The process remains an interprocess protocol between a server (IMS) and a number of clients (application programs). The client-adapter code that uses OTMA is responsible for obtaining and owning the context identifier (ID). In messages passed to OTMA, the context-ID field represents the sync_level=SYNCPT request. When IMS finds the context-ID in the message, IMS assumes the role of a participant in the two-phase commit process, as it does in the APPC environment.[6]

The IMS TM OTMA server function interfaces to the message queue resource manager and the EMH resource manager for delivery of messages to or from application program processing. Since the existing application programs use the DL/I call interface for their message processing requests, they continue to run transparent to the communication protocol supported by the OTMA client interface. The OTMA function opens IMS to access with other communication protocols. The OpenEdition DCE (data circuit-terminating equipment) Application Support server for IMS (AS/IMS) and the MQSeries* IMS bridge provide OTMA clients that interface with IMS.

The Internet arrives

With the increasing use of Transmission Control Protocol/Internet Protocol (TCP/IP) as a network protocol, IMS needed to enhance the existing TCP/IP interface support. Since IMS TM had already established the OTMA server function, IMS implemented the TCP/IP OTMA connection (ITOC) client function. The ITOC runs as an authorized MVS application program. ITOC provides support for user exit routines that can be used by TCP/IP clients to access IMS. ITOC defines a message prefix that is used to identify the user exit routine and a standard interface for calling the user exit routine.

The accelerating use of the Internet and the World Wide Web established the requirement for IMS Internet solutions. The IMS Internet solutions are called "IMS Connectors" and are a part of IBM's e-business plan to enable access to enterprise applications and data over the Internet using a Web browser. The ITOC is one of the four IMS Connectors that are provided for a communication access solution to IMS. The IMS Client for Java** Connector provides templates for preparing a Java applet or application program to access IMS application programs and data. The IMS Web Connector provides the ability to dynamically create transaction-specific programs and HyperText Markup Language (HTML) to enable Web browser access to IMS. The IMS Client for Java and the IMS Web Connectors use the ITOC for accessing IMS TM. The IMS WWW Templates is the fourth IMS Connector and uses the APPC protocol to communicate with IMS TM.

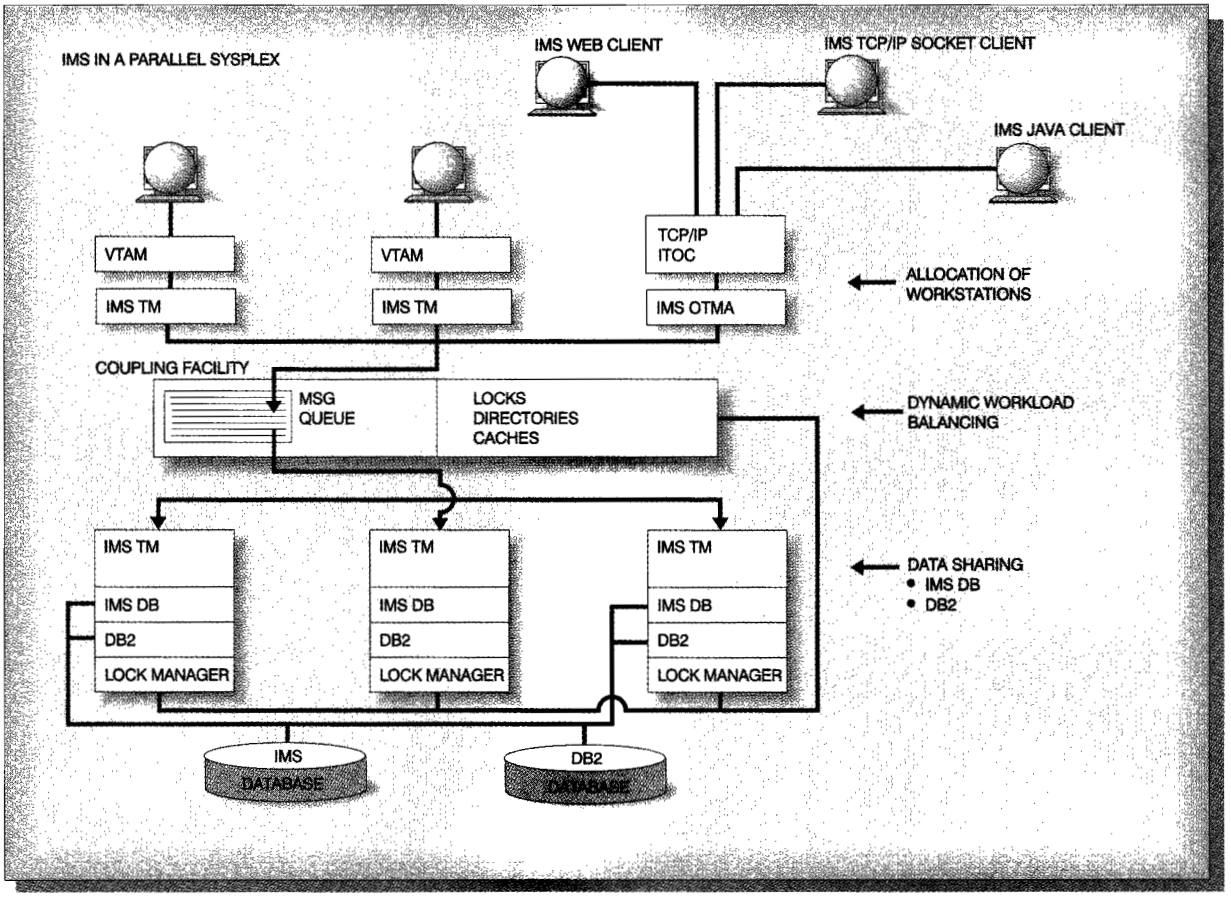
The interface to IMS TM for the four IMS Connectors for communication access enables DL/I message processing application programs to continue to process without change. The opening of access to IMS also requires the ability to access IMS DB.

The IMS Object Connector provides a new, easier way to access IMS DB managed data. New object-oriented application programs can use IMS Object Connector to access data managed by IMS. The object-oriented applications can be either IMS DB batch or IMS TM applications. Object-oriented application programs can also be invoked from a Web browser. The IMS Object Connector can use the IMS Open Data Base Access (ODBA) callable interface to access IMS DB. The ODBA connection interface enables any OS/390 application program to access the IMS DB full-function and fast path databases and to use RRS to provide the commit or backout coordination of database updates. In conjunction with the IMS birthday, IMS Version 6 is available to exploit the S/390 Parallel Sysplex (see Figure 2).

IMS in an S/390 Parallel Sysplex

In an S/390 Parallel Sysplex, IMS DB resource managers support block-level multisystem data sharing. The resource managers can also exploit the coupling facility cache structures for database buffer manage-

Figure 2 IMS Version 6 in an S/390 Parallel Sysplex



ment. This enhances the sharing of databases by multiple IMS systems. The IRLM lock manager uses the coupling facility lock structures to provide the block-level locking of IMS databases.

IMS TM supports message queue sharing by using the coupling facility list structures. Multiple IMS systems can access and process messages on the shared message queue. IMS provides the Common Queue Server (CQS) to manage the queues in the coupling facility list structures. CQS has a structured interface that enables other programs besides IMS to connect to it and process the shared message queues.[8] To fully utilize the Parallel Sysplex, IMS systems should be defined as clones. A cloned IMS system configuration is one in which the same application system resources (including transactions, programs, and databases) are shared among multiple IMS systems operating in a Parallel Sysplex configuration that is also sharing a common transaction workload. Transactions executing in such a configuration should have access to a common set of DL/I databases through IMS Nways or two-way data sharing.

The multisystem data sharing and message queue sharing provides the following capabilities when using the S/390 Parallel Sysplex:

- Automatic workload balancing—All IMS systems in the sysplex can automatically process workload by using shared queues. In prior releases of IMS, workload balancing was a user responsibility, requiring the use of MSC or APPC.
- Incremental growth—As workload increases, new IMS systems can be dynamically added to the sys-

plex to process the extra workload. This approach supports peak processing periods.
- Increased reliability—If any IMS system in the sysplex fails, any of the remaining IMS systems can process the workload in the shared queues. If one or more of the IMS systems requires a cold start, the contents of the shared queues are not affected.

These capabilities continue the evolution of IMS availability.

Summary

Celebrating 30 years as an IBM product, IMS has continued to evolve to exploit new technologies while preserving customers' application program investments. IMS is recognized as the world's premier transaction and hierarchical database server and manages the majority of the world's corporate data. Over 90 percent of the *Fortune* 1000 companies use IMS as their DBMS[1] of choice for fulfilling the requirements of performance, reliability, and availability. For more information on IMS visit the following Web site: http://www.ibm.com/ims.

Acknowledgment

I would like to thank all the people who work with IMS for their dedication to the product. Special thanks go to Barbara Klein, Peggy Rader, Don Streicher, Suzie Wendler, and Vern Watts for their comments and ideas that helped to significantly improve this technical note.

Cited references

1. From SHARE Session 1226 presentation (February 26, 1998).
2. *IMS/ESA V6 Administrative Guide: Data Base*, SC26-8725-00, IBM Corporation; available through IBM branch offices.
3. *IMS/ESA V6 Operations Guide*, SC26-8741-00, IBM Corporation; available through IBM branch offices.
4. *IMS/ESA V6 DBRC Guide and Reference*, SC26-8733-00, IBM Corporation; available through IBM branch offices.
5. *IMS/ESA V6 Administrative Guide: System*, SC26-8730-00, IBM Corporation; available through IBM branch offices.
6. *IMS/ESA V6 Administrative Guide: Transaction Manager*, SC26-8731-00, IBM Corporation; available through IBM branch offices.
7. *IMS/ESA V6 OTMA Guide and Reference*, SC26-8743-00, IBM Corporation; available through IBM branch offices.
8. *IMS/ESA V6 CQS Reference*, LY37-3730-00, IBM Corporation; available through IBM branch offices.

General references

Connecting IMS to the World Wide Web: A Practical Guide to IMS Connectivity, SG24-2220-00, IBM Corporation; available through IBM branch offices.

IMS: Why Migrate?, GC26-8912-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Administrative Guide: Data Base, SC26-8725-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Administrative Guide: System, SC26-8730-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Administrative Guide: Transaction Manager, SC26-8731-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Application Programming: Design Guide, SC26-8728-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 DBRC Guide and Reference, SC26-8733-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Operations Guide, SC26-8741-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 OTMA Guide and Reference, SC26-8743-00, IBM Corporation; available through IBM branch offices.

IMS/ESA V6 Release Planning, GC26-8744-01, IBM Corporation; available through IBM branch offices.

IMS/ESA Version 6 Guide, SG24-2228-00, IBM Corporation; available through IBM branch offices.

Accepted for publication May 25, 1998.

Kenneth R. Blackman *IBM North America Division, Advanced Technical Support—Dallas Systems Center, Santa Teresa Laboratory, 555 Bailey Avenue, San Jose, California 95141 (electronic mail: kblackm@us.ibm.com).* Mr. Blackman is a member of the IMS Advanced Technical Support Group of the IBM Dallas Systems Center. His primary focus is connectivity to IMS. He started working with IMS in 1971 as a systems programmer. In 1983 he joined IBM at the Santa Teresa Laboratory as an IMS developer. He has worked in the areas of APPC/LU 6.2 support in IMS, object-oriented technology, IMS WEB, TCP/IP connectivity to IMS, and Open Data Base Access to IMS DB. He is a graduate of the IBM Systems Research Institute and has received an IBM Outstanding Technical Achievement Award. He has also received the IBM Fourth Plateau Invention Achievement Award.

Reprint Order No. G321-5693.