DB2 TODAY

International Magazine for Regional DB2 User Groups

May 1993 Volume 2 Number 2

Special Insert
Kidder Peabody
DB2 User Survey

Craig S. Mullins

THE DOORS TO DB2

The OOT Wave has Arrived

"How Long Will this Baby Run?"

User Group Meeting Agendas


TSO
CICS
IMS/VS
CAF

The Doors to DB2

A DB2 application program is not limited to a specific technological platform. There are four choices of environments for developing DB2 application systems: TSO, CICS, IMS/VS, or CAF. Each of these environments is a door that, when opened properly, provides access to DB2 data. Of course, the options actually available to you will be limited by the availability of these environments at your shop. Each of these environments has limitations and provides advantages and disadvantages that I'll be discussing.

by Craig S. Mullins

It is not my intent to teach how DB2

FIRST though, let me emphasize that it is not my intent to teach how DB2 is accessed from these environments, but to help you understand each environment's strengths and weaknesses.

Fundamentals

Certain fundamental facts are true regardless of the environment in which your DB2 program will operate. Each DB2 program must be connected to DB2 via an attachment facility— the mechanism by which an environment is connected to a DB2 subsystem. Additionally, a thread must be established for each embedded SQL program that is executing. The thread will be used to send requests to DB2, data from DB2 to the program, and to communicate the status of each SQL statement after it is executed through the use of the SQLCA.

Now, let's explore the process that is followed when invoking a DB2 application program. First, the program is initiated and calls up the attach facility appropriate for the program environment. Following this, security will be checked (external MVS security, internal environment security and DB2 security). Finally, upon execution of the first SQL statement in the program, a thread is created. A thread is nothing more than a control structure used by DB2 to communicate with an application program. Every program, regardless of the environment in which it executes, must communicate with DB2 through a thread.

When the thread is established, DB2 will begin the process of loading the executable form of the plan (called a cursor table, or CT) into memory from the DB2 Directory. Plans are stored in the directory as a structure called a skeleton cursor table, or SKCT. If the plan is composed of packages, it will also load the package table (PT) from the directory SKPT. These CTs and PTs are loaded into an area of memory reserved for DB2 program execution called the Environmental Descriptor Management Pool, or EDM Pool. DBDs required by the plan will also be loaded into the EDM Pool from the DB2 Directory.

Now that we have a basic understanding of the way application programs communicate with DB2, let's explore the different processing environments for DB2. In the most basic terms, DB2 programs can be run in either foreground, otherwise called on-line; or in background, normally called batch.

On-line applications are characterized by interaction with an end-user via a terminal. Most on-line applications display a screen prompting a user for input, accept data from that screen, process the data and display another screen until the user determines that the session is to be terminated. On-line programs are generally used to provide real-time update and query capabilities, or to enter transactions for future batch processing.

Batch applications, by contrast, are characterized by their lack of user interactions. A batch program typically is submitted via JCL. It can accept parameters as input, but does not rely upon an end-user being present during its execution. Batch programs are generally used to perform mass updates, create reports and perform complex, non-interactive processes.

Each specific environment provides different modes of operation within each of these basic categories. For a quick overview of which environment supports which mode, consult Figure 1.

Overview of the available DB2 Processing Environments

Environment	Batch	On-Line
TSO	Y	Y
CICS	N	Y
IMS	Y	Y
CAF	Y	Y*

* CAF supports on-line programs when used in conjunction with TSO

Figure 1

TSO

TSO, the first of these environments, is short for Time Sharing Option. It enables users to interact with MVS using an on-line, optionally panel-driven interface. The Interactive System Productivity Facility, or ISPF, provides the mechanism for communicating via panels; it is the primary vehicle for users operating under TSO. Access to DB2 resources is provided via the TSO Attachment Facility in one of two ways:

☐ On-line, in TSO foreground, driven by application programs, CLISTs and/or REXX EXECs coded to communicate with DB2 and TSO, possibly using ISPF panels.

is accessed from these environments, but to help you understand each environment's strengths and weaknesses.

☐ In batch mode using the TSO Terminal Monitor Program, IKJEFT01, to invoke the DSN command and run a DB2 application program.

TSO is one of the three on-line environments supported by DB2, but unlike the other two, TSO is not transaction-driven. The TSO Attachment Facility operates on a communication channel that uses a single thread to direct DB2 calls. Only one thread can exist at any one time for each TSO address space, and each user can be logged on, in foreground, to only one TSO address space at any given time. Each batch TSO job, however, initiates a different invocation of the TMP, allowing numerous batch TSO jobs submitted by the same user to be running simultaneously. The batch jobs are independent from any foreground TSO activity, so a single user, at any given time, can have:

☐ One on-line TSO session communicating with DB2.

☐ Multiple batch TSO jobs communicating with DB2.

Simply by installing DB2, the TSO Attachment Facility is available for use. Communication between DB2 and TSO is accomplished via the DSN command processor—bundled with DB2—which enables DB2 commands to be issued in the TSO environment. One of these, the RUN command, is used to execute DB2 application programs. A sample is shown in Figure 2.

Additionally, IBM bundles two on-line TSO applications with DB2 that can be used to access DB2 data: DB2 Interactive (or DB2I, for short) and Catalog Visibility.

The DSN command processor establishes the thread that enables TSO to communicate with DB2. An alternative method is to use the Call Attach Facility within TSO to communicate with DB2. The Call Attach Facility will be discussed later in this article.

CICS

The second of the four "doors to DB2" is CICS (Customer Information Control System), a teleprocessing monitor that enables programmers to develop on-line, transaction-based programs. By means of BMS (Basic Mapping Support) and the data communications facilities of CICS, programs can display formatted data on screens to end-users and receive formatted data from end users for further processing. A typical scenario for the execution of a CICS transaction follows:

1) Operator enters data on terminal (including a transaction ID) and presses enter. This can simply be a transaction ID entered by the operator or a formatted BMS screen with the transaction ID.

2) Data is read into terminal I/O area and a task is created.

Sample DSN RUN Command

```
DSN SYSTEM (DB2P)
RUN PROGRAM (PROG0001) -
PLAN (PLAN0001) -
LIB( 'APPL.LOAD.LIBRARY' )
END
```

Figure 2

CICS must be attached to DB2 before any transaction can access DB2 Data. This is accomplished via the CICS Attach Facility.

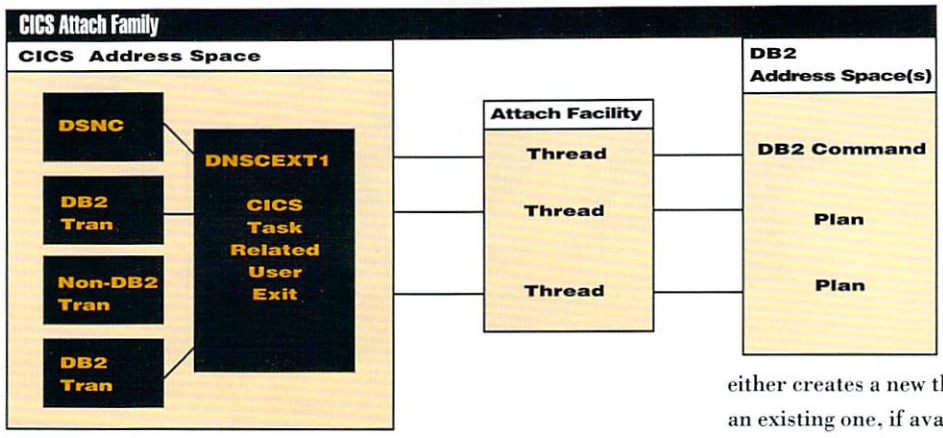

Figure 3

3) Checks to ensure that the transaction ID is valid.

4) If the program for this transaction is not in main storage, it is loaded.

5) Task is placed into queue until it is dispatched.

6) When the task is dispatched the appropriate application program will be run.

7) Program requests BMS to read data from the terminal.

8) BMS reads the data and the program processes it.

9) Program requests BMS to display data to a terminal.

10) BMS displays the data.

11) The task is terminated.

When DB2 data is accessed via CICS, multiple threads can be active simultaneously, enabling concurrent access to a DB2 subsystem by multiple users of a single CICS region. Contrast this with the TSO environment, where only one thread can be active for any given TSO address space. A mechanism named the CICS Attach Facility is used to connect CICS with DB2. Using the CICS Attach Facility:

☐ Any given CICS region can be connected to one and only one DB2 subsystem at one time.

☐ Any given DB2 subsystem can be connected to multiple CICS regions at the same time.

In order to fully understand how CICS controls the execution of an application program, one must first understand the relationship among tasks, transactions and programs. These three terms each define separate entities that function together, under the control of CICS, to create an on-line processing environment.

A task is simply a unit of work scheduled by the operating system. CICS, a batch job, DB2 and TSO are examples of tasks. CICS, however, can schedule tasks under its control much like an operating system schedules tasks. A CICS task, therefore, is a unit of work, composed of one or more programs, scheduled by CICS.

A transaction initiates a task. A CICS transaction is initiated by an identifier, 1 to 4 bytes long, defined to CICS via a control table. There is generally a one-to-one correspondence between CICS transactions and CICS tasks, but it is possible for one transaction to initiate more than one task.

Finally, a program is an organized set of instructions designed to accomplish some objective in a given unit of work. A CICS program can perform one or many CICS tasks.

As mentioned earlier, CICS must be attached to DB2 before any transaction can access DB2 data. This is accomplished via the CICS Attach Facility. The basic operation of the CICS Attach Facility is depicted in Figure 3. The CICS Attach Facility provides support for multiple transactions using multiple threads to access data in a single DB2 subsystem. CICS transactions requiring DB2 resources are routed to DB2 via DSNCLI each time an SQL statement is encountered, using the functionality of the CICS Task Related User Exits (TRUE).

The TRUE formats the request for DB2 data and passes it to the CICS Attach Facility, which either creates a new thread or reuses an existing one, if available.

When a thread is created the following activities will occur:

☐ A process known as DB2 sign-on is initiated, whereby the authorization ID— identifying the user of the thread— is established based upon a parameter specified in the RCT.

☐ A DB2 accounting record is written.

☐ Authorization is checked for the user.

☐ The executable form of the plan is loaded into memory. More specifically, if the header portion of the SKCT is not already loaded into the EDM Pool, it will be loaded at this point. The SKCT header will be copied into an executable form called a cursor table, and placed in the EDM Pool.

☐ If VALIDATE(RUN) was specified when the plan was bound, an incremental bind will be performed. This should be avoided.

☐ If ACQUIRE(ALLOCATE) was specified at bind time for the plan, locks for all tablespaces used by the plan will be acquired. DBDs referenced by the plan will be loaded into memory (EDM Pool), and all datasets to be used by the plan will be opened, if they are not already.

After the thread is created, the plan corresponding to the transaction being executed is allocated and the SQL statement is processed. When the request for DB2 resources is satisfied,

Continued on page 45

Continued from page 43

the data is passed back to the requesting CICS program via the TRUE. The thread is placed in an MVS-wait state until it is needed again. When the next SQL statement is encountered in the CICS program, the entire process is repeated except for thread creation, because the thread has already been allocated and is waiting to be used.

When the CICS task is terminated, or a CICS SYNCPOINT is issued, the thread is terminated, causing the following actions:

☐ The CICS Attach Facility performs what is called a Two Phase Commit. This functions to synchronize the updates and commits made to all defined CICS resources (ie., IMS databases, VSAM files, sequential files) and DB2 tables.

☐ A DB2 accounting record is written.

☐ Tablespace locks are released.

☐ The executable form of the plan is freed from the EDM Pool.

☐ Memory used for working storage is freed.

☐ If CLOSE(YES) was specified for tablespaces or indexes used by the thread, the underlying VSAM datasets will be placed on the drain queue for closing (provided no other resources are accessing them).

The CICS Attach Facility is started via the DSNC STRT command, indicating the specific RCT to use. The RCT (Resource Control Table) defines the attachment of CICS and DB2. It also assigns a type of thread to each CICS/DB2 transaction. There are three types of threads that can be used by CICS transactions to access DB2:

☐ ***Command Threads***, usable only by the DSNC command processor. If no command threads are available, then pool threads will be used instead.

☐ ***Entry Threads***, also called dedicated threads. These are associated with a single, specific application plan. Multiple transactions can be identified to an entry thread grouping defined in the RCT, but each transaction must use the same application plan. Entry threads are reusable by subsequent CICS transactions which use the same application plan. This can result in decreased run time because the cost of establishing the thread is avoided when it can be reused.

Entry threads can be defined as either protected or unprotected. A protected thread remains available for a pre-set time interval, waiting for transactions that can reuse the thread. Unprotected threads will be terminated upon completion unless another transaction is already waiting to use it. If an entry thread is not available for a transaction's use, it may be diverted to the pool, where it will utilize a pool thread.

☐ ***Pool Threads*** may be used by any transaction specifically defined to the pool. In addition, any transaction can be defined to be divertable. A divertable transaction is one defined to an entry or command thread where, if no appropriate threads are available, the transaction will be diverted to use a pool thread. A pool thread is not reusable and will always be terminated upon completion of the transaction using it.

IMS

IMS/VS is IBM's pre-relational database management system offering. It is based upon the structuring of related data items into inverted trees or hierarchies. The acronym IMS stands for Information Management System. Although usually perceived as a DBMS only, IMS/VS is actually a combination of two components:

☐ IMS/DB, the database management system.

☐ IMS/TM, the data communications environment, or teleprocessing monitor (also known as IMS/DC).

Either of the components of IMS can be used separately or together. On-line access to IMS databases can be achieved via either IMS/DC or CICS. Access to IMS databases is also provided in a batch environment. When an IMS database is accessed via IMS/DC it is said to be on-line; when it is accessed in batch it is said to be off-line.

IMS/DC provides an on-line environment in which application programs can be run that communicate with a terminal, much like CICS. Likewise, IMS/DC can be used by programs that access not only IMS databases, but DB2 tables as well. Although IMS and CICS are alike in many respects, the IMS environment differs from CICS in many significant ways. For example:

☐ IMS uses a facility called MFS (Message Format Services) to format messages to terminals and printers; CICS uses BMS.

☐ IMS/DC does not rely upon tables to control its environment, but on a series of macros known as a SYSGEN. The SYSGEN defines the terminals, programs, transactions and the general on-line environment for IMS/DC.

☐ All IMS programs require a program specification block (PSB) that defines the access to IMS/DB databases and IMS/DC resources. These PSBs are defined, along with IMS DBDs that define the structure of the IMS databases to be accessed, to control a program's scope of operation.

☐ An additional control block, the ACB (application control block), is used in the on-line world (and optionally in the batch environment) to combine the PSBs and DBDs into a single control block defining the control structure and scope of all IMS programs.

☐ The IMS on-line environment is composed of many different processing regions. All IMS/DC activity is processed through a region. There are two types of regions:

☐ A control region that manages IMS activity and processes commands.

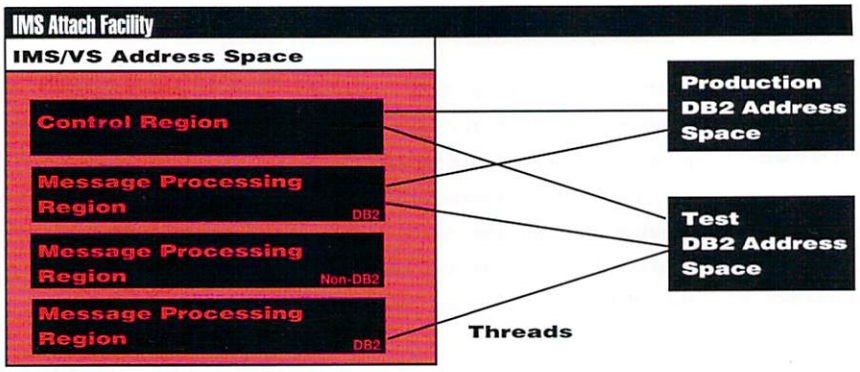

Figure 4

☐ **Dependent regions**, up to 255 per IMS/DC subsystem. Application programs execute from dependent regions.

IMS programs are categorized based upon the environment in which they run and the types of databases they can access. There are four types of IMS programs:

An **IMS batch** program is invoked by JCL and runs as an MVS batch job. IMS batch programs can access only off-line IMS databases, unless IMS Data Base Recovery Control (DBRC) is used. When DB2 tables are accessed by IMS batch programs, they are commonly referred to as DL/I batch. DL/I—Data Language/I—is the language used to access data in IMS databases, just as SQL is the language used to access data in DB2 tables. Batch DL/I programs run independently of the IMS/DC environment.

The second type of IMS program is a **batch message processor**, or BMP. BMPs are a hybrid program combining elements of both batch and on-line programs. A BMP runs under the jurisdiction of IMS/DC but is invoked by JES and operates as a batch program. All databases accessed by a BMP must be on-line to IMS/DC. There are two types of BMPs:

☐ Terminal-oriented BMPs can access the IMS message queue to send or receive messages from IMS/DC terminals.

☐ Batch-oriented BMPs do not access the message queue and cannot communicate with terminals.

True on-line IMS programs are called **message processing programs**, or MPPs. They are initiated by a transaction code, access on-line databases and communicate with terminals via the message queue.

The final type of IMS program is a **fast path program**—high performance MPPs that access a special type of IMS database known as a fast path database.

As with the other environments, a specialized attachment facility is provided with DB2 to enable IMS to access DB2 resources. The IMS attach facility, due to the nature of IMS, provides more flexibility for connecting to DB2 than the attach facilities for TSO or CICS. Refer to Figure 4:

☐ One DB2 subsystem can connect to multiple IMS subsystems

☐ One IMS subsystem can connect to multiple DB2 subsystems

☐ One IMS region can connect to multiple DB2 subsystems

☐ One IMS application program can access only one DB2 subsystem

DB2 is connected to IMS by a subsystem member (SSM). The SSM defines the parameters of the IMS attach facility for both on-line and batch connections.

CAF

The final "door to DB2" is the Call Attach Facility (CAF). CAF differs from the previous attach mechanisms since it does not provide teleprocessing services. CAF manages connections between DB2 and batch and on-line TSO application programs, without the overhead of the TSO terminal monitor program.

CAF programs can be executed in one of four ways:

☐ As an MVS batch job

☐ As a started task

☐ As a TSO batch job

☐ As an on-line TSO application

CAF controls a program's connection to DB2 as depicted in Figure 5. The DB2 program communicates to DB2 through the CAF language interface, DSNALI. The primary benefit of using CAF is that the application can control the connection using CAF calls. There are five CAF calls:

CONNECT	Establishes a connection between the program's MVS address space and DB2.
DISCONNECT	Eliminates the connection between the MVS address space and DB2.
OPEN	Establishes a thread for the program to communicate with DB2.
CLOSE	Terminates the thread.
TRANSLATE	Provides the program with DB2 error message information in the SQLCA.

Figure 5

Prior to DB2 V2.3, these calls had to be coded in Assembler because it was necessary to examine registers to ascertain return and reason codes. DB2 V2.3, however, eliminates this; it provides optional parameters to the CAF calls, enabling the retrieval of return and reason codes by high level languages.

Typically, a control program is created to establish and terminate the DB2 connection. This module is not required, but it is recommended to eliminate the repetitious coding of the tedious tasks associated with connecting, disconnecting, opening and closing in every CAF program.

CAF programs must be link-edited with the CAF language interface module, DSNALI.

CAF programs must be link-edited with the CAF language interface module, DSNALI.

The benefits of CAF are many. First and formeost, CAF provides explicit control of thread creation. Also, a program designed to use CAF can run when DB2 is down. It cannot access DB2 resources, but it will be able to perform other tasks. This can be useful when the DB2 processing is optional, parameter-driven, or contingent upon other parts of the program. And finally, CAF is more efficient than DSN, eliminating overhead required by the TSO TMP, IKJEFT01.

There are drawbacks to CAF, however. CAF requires more complex error-handling procedures. DSN automatically formats error messages for connection failures, whereas CAF will return only a return code and a reason code. Also, DSN automatically handles the connection; CAF requires the program to handle it. However, these drawbacks can be eliminated by coding this support into the CAF interface module at your site. This will involve your shop supporting logic—which is sometimes quite complex—that would otherwise be provided by DB2 itself.

Comparison of the Environments

Now that we have covered each of the environments in which DB2 programs can execute, we can begin to compare and contrast their features and capabilities. First and foremost, when choosing an operating environment for a DB2 application, one should ensure that it can support the data needs of the application. Typically, a corporation's data is spread across disparate processing platforms and data storage devices. Additionally, the data is stored in many different physical manifestations. When choosing an environment for your application, consider:

☐ Do I have access to the environment that I wish to use for a development platform? If not, can I obtain access?

☐ Is it possible to access enterprise data in its existing format, or will my choice of environment require that the data be duplicated and placed into a readable format?

☐ Are the programmers who will be working on the project knowledgeable in the chosen environment, or will extensive training be required?

Resource Availability

Figure 6 provides a breakdown of resource availability by processing environment.

Comparison of Resource Availability

Resource	CICS	TSO On-Line	TSO Batch	CAF TSO	CAF Batch	IMS MPP	IMS Fast Path	IMS BMP	IDL/I Batch
Flat File Access	Y	Y	Y	Y	Y	N	N	N GSAM	Y GSAM
VSAM Access	Y	Y	Y	Y	Y	N SHISAM	N SHISAM	N SHISAM	Y SHISAM
IMS Databases On-Line	Y	N	N	N	N	Y	Y	Y	N
IMS Databases Off-line	Y	N	N	N	N	N	N	N	Y
Invoked by JCL	N	N	Y	N	Y	N	N	Y	Y
Invoked by Transaction	Y	N	N	N	N	Y	Y	N	N
Invoked by CLIST or REXX EXEC	N	Y	N	Y	N	N	N	N	N
Invoked by ISPF	N	Y	N	Y	N	N	N	N	N

Figure 6

This chart can be used as a reference when choosing a processing environment for your DB2 applications.

Some of the entries in this chart for the different types of IMS programs may be confusing. The following list should clear up this confusion:

☐ The presence of a "Y" indicates that the processing environment listed across the top of the chart can access the resource defined along the left side. Although, the resource is accessible, it might not be possible to do so in your shop. Certain shops pose restrictions and limitations upon access; always consult your shop standards before proceeding with

Each environment has its own strengths and weaknesses; these should be surveyed before leaping headlong into an application development solution.

development plans based upon this chart.

☐ Flat file access is available using IMS calls when a GSAM database is defined for the flat file. (GSAM stands for Generalized Sequential Access Method.) IMS BMPs and batch programs can access flat files as GSAM databases. Access to flat files using pure OS/VS reads and writes is available to IMS batch programs only.

☐ All IMS programs can access VSAM KSDS datasets as a SHISAM database. (SHISAM stands for Simple Hierarchic Indexed Sequential Access Method.) Once again, IMS batch programs are the only type of IMS program that can access a VSAM file using VSAM dataset commands.

☐ IMS on-line databases are those defined to the IMS control region and started for on-line access within IMS/DC. Conversely, an off-line IMS database is either not defined to the IMS control region and hence, not accessible by IMS/DC or it is stopped (sometimes referred to as DBRed) to IMS/DC.

Feasibility

After ensuring that what is desired is possible, the next step is to ascertain whether it is feasible. It is feasible to develop an application in a specified environment if the response time and availability requirements of that application can be satisfactorily met by the environment. Typically, a service level agreement should be drawn up for each new application that includes a price/performance matrix which end users must agree to. For example:

Example

The on-line portion of the system must provide an average response time of x seconds, y% of the time, for an average of z users. The cost per transaction will be approximately a.

Utilize the chart in Figure 7 to determine which on-line environment is feasible for any given project. Take the following items into account:

☐ What is the deadline for system development? What programming resources are available to meet this deadline? Do I have the requisite talent to develop the system in the environment that is optimal? If not, should I obtain them or settle for a less than optimal solution?

☐ What are the performance requirements of the system? How many concurrent users will be using the system during

Comparison of On-Line Development Capabilities

Capability	TSO	CICS	IMS/DC
Response Time	Slow	Fast	Fast
Flexibility	High	Low	Low
Number of Concurrent Users	Less than 10	Many	Many
Overhead per User	Very High	Very Low	Low
Program Linking	Not easy	XCTL LINK	Message Switching
On-line Screen Language	ISPF Dialog Manager	BMS	MFS
Screen Development	Fast	Cumbersome	Cumbersome
Program Development	Fast	Medium	Slow
Prototyping & Testing Tools Available	Many	Some (more than IMS)	Few

Figure 6

the peak processing time, and can the given environment support the workload?

Sometimes there is little or no choice: some shops have only one of the environments and for them, the decision is easy. However, if you have more than one environment, the right decision is never to use only one of them. Each environment has its own strengths and weaknesses; these should be surveyed before leaping headlong into an application development solution.

However, when multiple environments are used to access DB2 data, they become inextricably wound together into a critical mass. This can be difficult to administer and warrants consideration.

The Critical Mass

When accessing DB2, the teleprocessing monitor—be it TSO, CICS or IMS/DC—must reside on the same MVS system as DB2. This creates what is referred to as a *critical mass*. The critical mass is the set of subsystems tied together by a single common attribute: they must access DB2 resources. For example, if a data processing shop uses both CICS and IMS/DC to develop DB2 applications, their critical mass consists of the following:

- ☐ The IMS/DC subsystem
- ☐ All CICS subsystems requiring DB2 access
- ☐ The DB2 subsystem
- ☐ A TSO subsystem if DB2I access is required

These must all operate on the same CPU. Additionally, in an error situation, they can not be moved independently without losing DB2 access. One can see where a large shopmight quickly tap the resources of its machine if all DB2 applications are developed on a single DB2 subsystem. To avoid this situation, consider slicing applications into disparate, independently operating units by:

- ☐ Developing IMS/DC applications on one DB2 subsystem, CICS applications on another, and TSO application on yet another. This will reduce the critical mass such that IMS/DC and CICS are not married together.
- ☐ Providing distributed access between the separate DB2 subsystems for access to DB2 data which must be shared among applications. Read-only access from one DB2 subsystem to another is provided, without requiring a distributed query, as it is with DB2 V2.3.
- ☐ If at all possible, choose a single teleprocessing environment for all DB2 applications.
- ☐ By avoiding DB2I and QMF access, TSO can be eliminated from the critical mass. You can submit SQL and DSN commands as batch invocations of TSO, but this hampers ease of use and detracts from the overall user-friendliness of DB2. As such, it is not recommended.

Synopsis

Each of the four environments that can access DB2 data provides both strengths and weaknesses. Hopefully, this article has helped you to determine which of the four environments will be suitable for your different types of application development efforts. ■

TSO

CICS

IMS/VS

CAF

Craig S. Mullins works in the Technical Advisory Group at PLATINUM technology, inc. *His first book,* DB2 Developer's Guide, *published in 1992 by Prentice-Hall Computer Publishing, contains , more than 1100 pages of DB2 development techniques, tips and guidelines.*