IBM Database 2 in an Information Management System environment

by J. R. Dash
R. N. Ojala

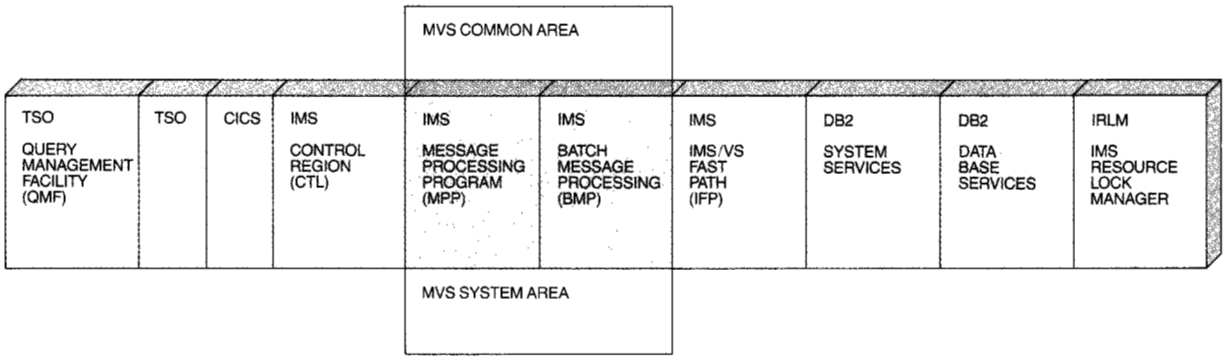
Over the years, the IBM Information Management System (IMS/VS) has been developed to meet expanding user needs. During that time, a parallel development has taken place. The relational data model grew from Codd's original theory to a practical data base prototype. Now a new data base management system, IBM Database 2 (DB2), has been built on the relational model. This paper discusses the implementation and design considerations for the integration of IMS and DB2 from the user's viewpoint. It also presents the attachment facilities from a design perspective.

The IBM Information Management Systems, IMS/VS, had its origins in IMS/360, which was announced in 1969. Over the years since its first development, IMS has greatly improved in function to support the rapidly growing need for data-communications-based applications.[1] The data base part of IMS/VS, known as IMS/DB or DL/I (Data Language/I), is a hierarchical data base management system (DBMS). In a hierarchical or tree data structure, each application threads its way from data record to data record accessing groups of fields called *segments*. The connection between data records is established via pointers. IMS/VS is referred to in this paper simply as IMS.

In the late 1960s and early 1970s, Codd[2] introduced the relational data model as an alternative way of structuring and managing data. Here, data are structured in two-dimensional tables and related by their value only, not by pointers embedded between data records. In the lexicon of data base terminology, this structure has created the term *nonnavigational data structure* because no programmer navigation is required to move through the data structure. In conjunction with the data structure, the relational model suggests data manipulation via a series of set-theoretic operators that help achieve significant economies in programming and end user access to data bases. The new MVS data base management system IBM Database 2 (DB2) is built upon the relational data model. Overall design principles of DB2 are discussed by Haderle[3] and Kahn[4] elsewhere in this issue.

This paper discusses the linking of DB2 with IMS (IMS/VS, Version 1, Release 3). IMS application programs can retrieve and update data in DB2 tables using the facilities discussed in this paper. Whereas some application programs may access DL/I data bases only, other programs may access both DL/I and DB2 data, or perhaps access DB2 data only. Because the DB2 system can be used by the Customer Information

Figure 1 MVS address spaces



Control System (CICS), Time Sharing Option (TSO), and Query Management Facility (QMF) users on the same host, data gathered by one system can be shared easily by all systems within the same host. Figure 1 shows various MVS address spaces that may be used with IMS.

We treat the subject of DB2 in the IMS environment principally in two parts. We first describe various aspects of DB2 in IMS from a user's point of view. We then look at the DB2-IMS attachment facilities from a systems design perspective.

Use of DB2 in an IMS environment

Users of DB2 in an IMS environment comprise application analysts and programmers, data base designers and administrators, systems programmers, security administrators, operators, and end users. While end users of IMS transaction programs do not see any difference with DB2, business professionals will find DB2 and QMF to be effective tools for access to data extracted from IMS production data bases.

Data base design. From a theoretical viewpoint, the design of a data base on a logical level should not be different for different data base management systems. The logical data base design should resolve questions as to which records are best suited to model the real business, which fields should go together in a record type, when it is better to split a record type into multiple record types, and what relationship exists between record types. Data base design theories covering these steps are somewhat formalized in the process termed *normalization.*[5-7] Normalization guidelines are designed to reduce update anomalies and data inconsistencies. These guidelines are not exclusive to relational data base systems; they apply as well to the design of record structures for such nonrelational DBMSs as IMS/DB.

After the normalization process, there follows the physical design task to structure these record types into physical data bases according to the requirements, facilities, and peculiarities of the actual DBMS. Although logical data base design should be independent of the physical aspects of data structure, usually the boundary is not as clean and distinct as it should ideally be. During the logical design phase, an experienced designer should take into account the effect of options on the physical design. At least, this had been the way the designer would proceed, until the appearance of relational data base systems.

An important characteristic of relational DBMSs is the enhanced level of data independence (i.e., the separation of the user's view of data from its storage representation). Data are always modeled as values in actual tables. Therefore, many complex considerations are eliminated, such as which relationships should be implemented in the data structures (hierarchies and networks) and which should still be implemented as values (i.e., the decision between a direct and a symbolic pointer in DL/I). This leads to a simplification in going from a logical design to a physical design.

In summary, the results of the logical design of a set of record structures map directly into a set of DB2 tables. The advantage of this type of design activity is that DB2 table structures have a better chance to survive changing requirements and needs. This is in

contrast to a network or hierarchical system, where there is the further step of deciding which relationships should be implemented as part of the data structure.

Data base implementation. When the resulting data base design is implemented and defined to DB2, a number of tasks are simplified. Logical objects are clearly separated from physical ones. The data base administrator can clearly define physical aspects (e.g., which indexes, partitioning or not, page size, or single or multiple tables per tablespace), and application designers can provide such logical definitions as tables and views. Definitions of data base structures can be accomplished directly from an online terminal and with no disruption to current data base operations. An example of the SQL CREATE statement for an EMPLOYEE table is shown in Figure 2 and for a DEPARTMENT table in Figure 3. Data base definitions take immediate effect in DB2 because the Data Definition Language (DDL) statements are processed instantly, resulting in updates to the DB2 catalog. Additional columns can be added without requiring unloading and reloading of the data base by using the SQL ALTER statement. View definition may insulate one design from subsequent changes. DB2 is more forgiving of an incomplete data base design. Thus, it is not so critical to make the design right the first time.

DB2 offers a uniform language (SQL) for both data base access and definition. Having one consistent language for these two tasks simplifies communication among persons who define the data bases and those who will later access them. The uniformity of language for data definition and data access has also contributed to the fact that with SQL some aspects of data base programming have been moved over to data base definition. For example, the view definition (D01ADMIN) shown in Figure 4 eliminates the need for program code that otherwise would have been necessary in order to combine the EMPLOYEE and DEPARTMENT tables and to obtain the proper subset of rows.

Finally, among the task simplifications, DB2 provides an authorization mechanism that allows data base administration to be centralized, partially decentralized, or completely decentralized. Portions of the data base administration function can be parceled out to different areas without giving up complete control.

Application programming. An application program in the IMS environment can have accesses to DB2 data intermixed with accesses to DL/I data. Communication with an IMS terminal uses the DL/I mechanism. Transaction input is received using the DL/I, GU, or CHKP call, and messages are sent to terminals using ISRT or PURG calls. Applications running in the IMS batch region have no access to DB2 tables. Applications can, however, run as Batch Message Processing (BMP) programs to communicate with DB2. The IMS end user at the terminal should note no differences if a transaction accesses DB2 data. No messages relating to DB2 are sent to the end user by IMS.

The IMS end user should note no differences if a transaction accesses DB2 data.

While details of application programming techniques can be found elsewhere,[8] we describe here some of the concepts and functions in DB2 that are new to the IMS programmer. Because relationships in DB2 tables are represented only by values in the tables themselves, there are some significant advantages. At the user level all relational access is accomplished through associative addressing by comparing values. Data base requests to retrieve or update data are always made through the use of actual values in the tables. This simplifies operations on tables by not requiring data base requests in terms of an implied position, as in DL/I structures, and a movement along the segments within the structure.

A consequence of access through field values only is that all access is symmetrical. The programmer does not code differently depending on what access path is used. Access through different indexes, as an example, is not different in the way one codes, as in IMS/VS. The use of an index is determined completely at the internal system level and is not specified by the user in DB2. Therefore, indexes can be added or deleted without affecting program logic.

The SQL language can be used as a stand-alone language (via SPUFI in TSO/SPF), or it can also be embedded into conventional host languages such as

Figure 2 Employee table showing definition statement and table contents after LOAD

```
CREATE TABLE EMPLOYEE
   (EMPNO       CHAR(6)       NOT NULL,
    LASTNAME    VARCHAR(15)   NOT NULL,
    WORKDEPT    CHAR(3)       NOT NULL,
    PHONENO     CHAR(4),
    JOBCODE     DECIMAL(3),
    EDUCLVL     SMALLINT,
    SEX         CHAR(1),
    SALARY      DECIMAL(8,2));
```

EMPNO	LASTNAME	WORKDEPT	PHONENO	JOBCODE	EDUCLVL	SEX	SALARY
000100	Spenser	E21	0972	54	14	M	26150
000140	Nicholls	C01	1793	56	18	F	28420
000260	Johnson	D21	8953	52	16	F	17250
000310	Setright	E11	3332	46	12	F	15900
000030	Kwan	C01	4738	60	20	F	38250
000060	Stern	D11	6423	55	16	M	32250
000010	Haas	A00	3978	66	18	F	52750
000220	Lutz	D11	0672	55	18	F	29840
000300	Smith	E11	2095	48	14	M	17750
000020	Thompson	B01	3476	61	18	M	41250
000050	Geyer	E01	6789	58	16	M	40175
000070	Pulaski	D21	7831	56	16	F	36170
000270	Perez	D21	9001	55	15	F	27380
000150	Adamson	D11	4510	55	16	M	25280

Figure 3 Department table showing definition statement and table contents after LOAD (where < > indicates null value)

```
CREATE TABLE DEPARTMENT
  (DEPTNO      CHAR(3)      NOT NULL,
   DEPTNAME    VARCHAR(36)  NOT NULL,
   MGRNO       CHAR(6),
   ADMRDEPT    CHAR(3));
```

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
A00	Spiffy Computer Co	000010	< >
B01	Planning	000020	A00
C01	Info Center	000030	A00
D11	Manufacturing	000060	D01
D01	Dev Center	< >	A00
D21	Admin Systems	000070	D01
E01	Support Services	000050	A00
E11	Operations	000090	E01
E21	Software Support	000100	E01

< > indicates NULL value.

COBOL, PL/I, or Assembler. This is an important requirement for a relational language like SQL, and this property is characterized as the *uniformly relational property.*[9]

SQL operations apply to entire collections (sets) of rows from tables, not just individual rows. The result of each table operation is itself a table. Therefore, data base requests can be expressed so that results of one operation become the inputs to another. This set-at-a-time capability is extremely powerful compared to record-at-a-time processing, where the coding is required to be more detailed. This power is fully available to the interactive user of SQL.

Figure 4 D01ADMIN view showing employees and departments reporting to Department ID D01

```
CREATE VIEW D01ADMIN AS
 SELECT EMPNO,LASTNAME,DEPTNO,DEPTNAME
 FROM EMPLOYEE,DEPARTMENT
 WHERE WORKDEPT = DEPTNO
 AND ADMRDEPT = 'D01'
```

EMPNO	LASTNAME	DEPTNO	DEPTNAME
000150	Adamson	D11	Manufacturing
000220	Lutz	D11	Manufacturing
000060	Stern	D11	Manufacturing
000260	Johnson	D21	Admin
000270	Perez	D21	Admin
000070	Pulaski	D21	Admin

For the embedded version of SQL (e.g., in COBOL and PL/I), the operations of insert, delete, and update present no problems as set operations. Also, there is no problem when retrieval operations return one row as a result. However, when retrieval requests return a set of rows to the application program, those rows are accessed one at a time by using a cursor. This is because the number of rows to be returned is not known when the SQL statement is issued. The cursor is a position holder in a set of rows, and this limits the value of set operators in the host language environments to some degree.

A single SELECT statement in DB2 (SPUFI) corresponds to a series of GU, GN, and GNP calls in DL/I. SELECT statements using join conditions correspond to retrieval calls, depending on links carried in the hierarchical structure (i.e., GNP and GN).

Data consistency facilities are generally a characteristic of the implementation. With the current implementation of DB2, consistency of data across tables in general has to be maintained by the programs and procedures that operate on the data bases. To a certain extent, this responsibility is simplified by the language functions available in SQL. IMS/VS provides some data consistency capabilities through its hierarchical data structures and other capabilities through the logical insert/replace/delete rules.

Primary key consistency in DB2 is accomplished by specifying that a primary key column is not allowed

to have null values and by specifying a unique index for the column. IMS/VS requires keys to be present and to be unique if so specified, but there are options available to depart from this.

DB2 provides built-in functions for summation (SUM), ordering (ORDER BY), and basic statistics [Minimum (MIN), Maximum (MAX), Average (AVG), and COUNT]. There are no corresponding functions in IMS/VS, and all such functions have to be coded in application programs.

DB2 provides a layered approach to security.

View definition can substitute for a considerable part of programming by eliminating irrelevant data that otherwise would have to be handled by program code. The example in Figure 4 illustrates this point. Also, if the same set of data is used in more than one application, redundant coding can be eliminated by establishing predefined view specification. The SQL CREATE with CHECK OPTION for single-table views can be used to protect against inserts or updates that might destroy the very criteria on which the view is based.

When program development is in a preliminary or early stage, it is convenient to try out relational data base request sequences directly from a terminal as a stand-alone data base test without involvement of an actual program. It is also convenient to set up and test data outside the actual program through the interactive terminal interface. This is possible because SQL is self-contained; i.e., it does not require a host language in order to be used. For IMS users, Batch Terminal Simulator (BTS) can also be used for testing application programs containing SQL statements.

Relational data bases offer the potential for additional benefits in data base design and application development. One such area is frequently termed *semantic modeling.* This refers to the incorporation of meaning into the formal structure of the data base, that is, meaning understood by the system. An example of a proposal based on the relational model is Codd's RM/T work.[10]

Data base security. Different approaches to security are used by IMS and DB2. DB2 provides a layered approach to security that allows the installation to distribute control of various functions to different organizational units which may then distribute control of certain functions to subunits. IMS installations can take advantage of such a scheme for distributed authorization. Data base security in DB2 is enforced by two approaches: the view mechanism and the GRANT/REVOKE facility.

View definitions are a powerful facility for simplifying programming. A view need contain only the data that a particular application has to deal with. From a security viewpoint, this facility shields those data that are irrelevant to a specific user. Security by content, in this way, is not specified in procedural program code but is available through the data definition facilities. Moreover, SQL built-in operators may be used in views to restrict some users to such aggregate data as averages and totals, without allowing them to access individual values.

GRANT/REVOKE commands are available to restrict users to performing specific operations against tables or views. The command GRANT INSERT ON TABLE EMPLOYEE TO PROG01 allows an application programmer to insert rows into an existing table. An authorization may be granted to a user WITH GRANT OPTION, thus allowing the user to pass on this authority to other users. Authorization can be restricted to specific operations on specific tables, columns, and rows.

The SQL REVOKE statement is used to take away a certain capability from a DB2 user. For example, the command REVOKE UPDATE ON DEPARTMENT FROM PUBLIC illustrates the revocation of the UPDATE authority from all users (PUBLIC) on the DEPARTMENT table, where the system administrator had previously granted this authority to all users.

Data base maintenance. After data bases acquire production status, changing requirements sometimes demand modifications to data base structures. The more frequently a data base structure changes, the more expensive the maintenance activities become.

The reasons for these increased maintenance costs include requirements (1) to change definitions, (2) to run certain utilities, and (3) to make program modifications. Changes to DL/I structures, such as adding a new segment type, establishing new logical relationships, changing pointer schemes, and adding a new secondary index, can require significant maintenance activities.

In DB2, maintenance caused by data base changes has been simplified.

In DB2, maintenance caused by data base changes has been simplified. In one way, changes have a lessened effect because all relationships are carried through actual values in the tables and not through links in an explicit data structure. Many changes to tables do not require data bases to be unloaded and reloaded. For example, new columns can be added using the SQL ALTER TABLE statement in an interactive session in TSO.

Also, indexes can be added to an existing table dynamically, using the SQL CREATE INDEX in the DB2 Interactive environment (index-on-the-fly). There is no requirement for running special utilities in an offline mode against the affected data bases.

View definitions can preserve a previous view of a table against new column additions.

The running of data base maintenance utilities from DB2 Interactive (DB2I) eliminates the need for scheduled outage of production data bases. The partitioning facility helps improve availability as does the AREA facility in fast path data bases in IMS.

User data base recovery. The basic principle of data base recovery is to maintain duplicated data. Duplication of the data base as well as the changes made to it is required. This principle is true for both IMS and DB2. In IMS, users periodically duplicate the data base (image copy) as well as changes via the DL/I log facility. Forward recovery is used for both logical and physical errors, whereas backward recovery (undoing changes made to the data base) helps solve logical errors in the data base.

The backup and recovery procedure is subject to error, because the user must keep track of all image copies, logs, and change accumulations. A very useful tool in IMS is the Data Base Recovery Control (DBRC), which keeps track of various data sets used and creates JCL for the various jobs needed in the backout and recovery process.

The equivalent of the DBRC function in DB2 is provided by the DB2 catalog and the Bootstrap Data Set (BSDS). The various backup and recovery utilities for DB2 are the following: Image Copy (full or partial), Incremental Image Copy, Merge Copy, and the Recovery utility. The DB2 log becomes the nucleus for the recovery process. Concepts of active DASD logging and automatic archiving are incorporated. Figure 5 illustrates the recovery scenario with the associated data sets. The DB2 backup and recovery concept allows the recovery of a table space, a partition, or a group of pages. Two utilities are provided to maintain image copies of tablespaces. The recovery utility may be used to recreate a tablespace from an image copy and also to log records. Some differences between DB2 and IMS are the following:

- In IMS the base unit for recovery is a data base data set, whereas in DB2 the base unit of recovery is a tablespace or part of a tablespace.
- In IMS without DBRC it is possible to do a partial recovery, i.e., restore a data base data set from an image copy without applying logged changes. This is sometimes used for regression tests. In DB2 (and in IMS/VS, Version 1, Release 3) a recovery consists of restoring a data base data set and tablespace from an image copy. Thereafter, all logged changes are applied.
- In IMS, index data sets have to be recovered separately, whereas in DB2, indexes are not recovered but rebuilt by using the DROP and CREATE INDEX statements of SQL.
- In DB2, there is no equivalent for the IMS change accumulation utility where data base change records from various log tapes are merged. The Merge Copy utility of DB2 is used to merge Image Copies and Incremental Image Copies (IIC) to construct a new full Image Copy or a composite IIC.
- There is no batch backout utility in DB2 as there is in IMS. If dynamic backout fails, it is necessary to perform a data base recovery or repair.

End user access to data. One of the major advantages of the relational model is the ability of the end user to perform unanticipated queries. End users have two ways of accessing DB2 data: (1) using DB2 Interactive (DB2I), or (2) using Query Management Facility (QMF). DB2 users in TSO/SPF can choose DB2I and then select SPUFI for an interactive session with DB2 data. SQL statements can be issued for an instantaneous response of results on the screen. QMF provides a more powerful end user facility to do specialized reporting and aggregating. Users can also specify report formats and store them along with the queries for repetitive work.[11]

Current IMS users can use the Data Extract (DXT) product to selectively extract from DL/I data bases existing production data (as well as VSAM and SAM data) and load them into DB2 format for access via DB2I or QMF. DXT also takes advantage of the information in the IBM DB/DC Data Dictionary in preparing for the extract process. The extract approach is useful where enterprises need to avoid end user access directly to production data for performance reasons.[12]

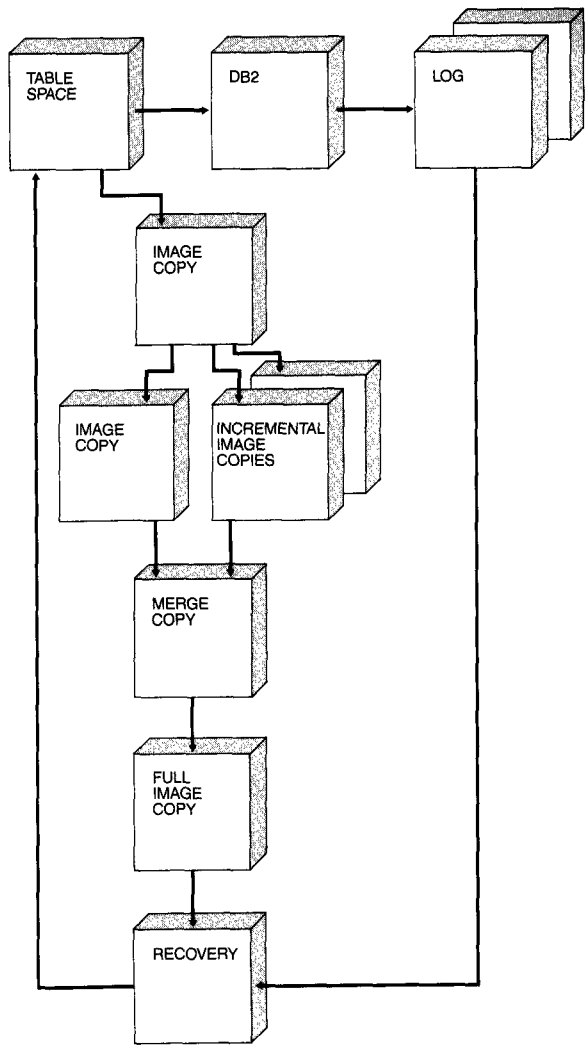
Existing IMS production programs can be modified to update DB2 data bases with summary information. The update may take place synchronously, by having the program update the data base, or asynchronously, by sending the information to a background IMS transaction that can in turn update the DB2 data base.

DB2 and IMS attachment facilities from a design perspective

So far we have been describing the use of DB2 in an IMS environment. In the remainder of this paper we discuss how DB2 and IMS have been designed to work together and what IMS installation personnel must understand in order to use DB2. The MVS address spaces used with DB2 are shown in Figure 1.

Attachment overview. DB2 and IMS are connected via an attachment package. The designers of the DB2–IMS attachment package carefully considered many design tradeoffs. The design chosen stresses ease of use and flexibility in preference to requiring an installation to predefine all DB2-related items to IMS. The attachment between existing DB2 and IMS systems can be installed without an IMS SYSGEN. Also, the attachment package makes use of numerous defaults to simplify installation and maintenance. Once IMS and DB2 are operational, they connect automatically and process DB2 requests from IMS without operator assistance. Operator commands are provided to monitor the status of the connection between on-line message regions and DB2. In addition, records are produced that allow the installation to monitor and account for IMS-related activity with DB2. The capability exists for multiple DB2s to connect to a single or multiple IMS systems in one host.

Figure 5 Data base recovery cycle



Defining DB2 to an existing IMS system. We now describe the steps and considerations that installation personnel must understand in order to install and use DB2 in an IMS environment. When DB2 and IMS

have been installed, the installation personnel perform the following three steps:

- Define a subsystem member in IMS.PROCLIB. That member specifies the DB2 system(s) that can connect to a specific IMS.
- Place the name of the subsystem member on the execute statement of the IMS control region JCL procedure.
- Place a DFSESL DD card specifying the DB2 library into the IMS control and each dependent region JCL.

The default condition allows any IMS region to process DB2 transactions. The attachment code resides in each IMS region and is loaded at message region initialization. For Batch Message Processing (BMP) regions, the attachment code is loaded when an application program issues the first SQL call. The installation may control which message regions process DB2 transactions by using IMS class scheduling.

An IMS application program may access and change IMS and DB2 resources.

Operating IMS with DB2. IMS and DB2 are both started by an MVS operator command. They can be started in any order, and they automatically connect when both systems are operational. No additional MVS or IMS operator action is required to activate the connection between the systems for normal operations.

Once the connection is operational, the connection may be stopped by an IMS or DB2 command, either in *quiesce* or *force* manner. Quiesce allows application programs accessing DB2 to terminate before the connection is broken. The STOP DB2 MODE(FORCE) command causes work in progress to be abnormally terminated. The DB2 stop force command can be used after a quiesce request. A wait-for-input BMP or other programs may be connected to DB2, but they may be waiting for another message or issuing IMS requests. The STOP DB2 MODE(QUIESCE) command request waits for these programs to terminate. It may be necessary to terminate connected IMS regions via an IMS stop region command. The DB2 stop force command can also be used to break the connection.

IMS provides commands to control and monitor the connection between IMS and DB2. There are also DB2-provided commands to control and monitor external connections. The IMS command /SSR allows IMS operators to issue DB2 commands. A DB2 command entered by an IMS operator must pass both IMS and DB2 security checking. The automated operator facility of IMS may also be used to submit DB2 commands from IMS.

Security considerations between IMS and DB2. A transaction scheduled by IMS must pass IMS security checks first and then go through the DB2 authorization, if an SQL call is issued. The security identification used by DB2 depends on the security specified for IMS.

For IMS message-driven programs, the authorization ID is the sign-on name if IMS is using sign-on authority. Otherwise, the authorization ID is the logical terminal name. For IMS non-message-driven programs, the authorization ID is the name of the user specified on the JOB card, if the Resource Access Control Facility (RACF) is present. Otherwise the Program Specification Block (PSB) name is used.

Coordinated recovery. An IMS application program may access and change IMS and DB2 resources. IMS and DB2 have coordinated recovery, which means that the status of changes to IMS and DB2 resources made by an IMS application program is consistent. The changes are either committed or aborted in both IMS and DB2. If a failure occurs in the IMS application program, in the IMS subsystem, in the DB2 subsystem, or in MVS, the resources involved are placed in a consistent state. DB2 provides utilities to recover from disk media problems. The utilities are similar in function, but operate independently of the IMS utilities, as has already been discussed in the section on user data base recovery.

Application design considerations. An application program issuing SQL calls must be precompiled using the DB2 precompiler to create a Data Base Request Module (DBRM). The DBRM is the input to the DB2 BIND process which produces a DB2 application plan (or simply plan) that contains an optimized access path for each SQL statement. Besides the access path,

the plan also contains the tables to be accessed and the appropriate locking information. The application program must be compiled and then link-edited with the IMS Release 3 language interface module.

When IMS schedules an application program, the needed IMS resources are available. The DB2 resources, however, may or may not be present, because the connection between an application program and DB2 occurs only when the program issues the first SQL call. The DB2 plan name, which is associated with the application program, is passed by the IMS attached to DB2 at this time. It is possible that the DB2 subsystem is not operational or the plan requested is not valid. To handle such possibilities, the installation can allow three options to the application program. The options, described in the following paragraphs, can be specified for the entire IMS system, on the basis of IMS-dependent regions, or on the basis of application program load modules.

The first option (the default) is for the application program to receive an SQL return code indicating the error. This allows the application program to notify the terminal operator that an error has occurred.

The second option is to back out any activity the program might have performed against DL/I, requeue the message, and prevent the transaction from scheduling until the problem is fixed. The transaction must be restarted by the operator when the problem is fixed. This option is similar to what IMS does when a PSB or IMS data base is not available. The message is not lost.

The third option is to abnormally end the program, back out any DL/I activity, and discard the message.

Also, at the first SQL call, DB2 authorization checking is performed. If the user is not authorized, the application program receives an SQL error code indicating the authorization error.

Currently in DB2, the size of a DB2 plan is directly proportional to the number and complexity of SQL statements in that application program. Also, DB2 obtains locks on the resources specified in the plan. Thus, it is recommended to have small-size programs performing specific functions.

DB2 uses the IMS Resource Lock Manager (IRLM) for locking. IMS can use program isolation locking or IRLM locking. An application program referencing both DL/I and SQL resources can end up with the following situations regarding deadlock detection:

- A true deadlock may be detected between multiple programs referencing DL/I resources.
- A true deadlock may be detected between multiple programs referencing DB2 resources.
- A potential deadlock between multiple programs referencing both DL/I and DB2 resources is detected only by an IRLM time-out. The time-out value is a DB2 installation parameter, and, if set too small, that value may give false deadlock conditions. The installation should determine whether application programs will access both DL/I and DB2 data bases and set the time-out value appropriately.

IMS and DB2 monitoring. IMS provides information about DB2 via the Data Communication (DC) monitor. The new monitor records contain information about each call to DB2, including the time when the call started and when it ended.

DB2 is designed to exploit the 31-bit addressing architecture of MVS/XA.

DB2 provides a number of tools that monitor storage space, tables, and application usage. For each new message processed by IMS, or at the termination of an application program, a DB2 accounting record is written to SMF. DB2 accounting keeps track of elapsed time, CPU time, SQL calls, locking information, buffer pool information, application program termination status—all on a user basis. The record also contains the user ID (authorization ID), the DB2 plan name, the IMS Partition Specification Table (PST) number, the IMS PSB name, the IMS subsystem name, and a time stamp. The installation can use these tools to determine the performance of IMS application programs accessing DB2 tables.

MVS/XA considerations. DB2 is designed to exploit the 31-bit addressing architecture of MVS/XA, but it can also run on MVS/370, which uses 24-bit addressing. The MVS/370 24-bit addressing range is up to 16 megabytes, whereas the 31-bit addressing range of MVS/XA is extended to 2 gigabytes. Programs operating in 24-bit addressing mode can access information up to the 16-Mb line only.

The majority of DB2 code and control blocks in MVS/XA have moved above the 16-Mb line, and the majority of DB2 Common System Area (CSA) usage has moved to the extended CSA above the 16-Mb line. Thus, a DB2 MVS/XA system uses less CSA space below the line than a comparable non-MVS/XA DB2 system. Another benefit of running DB2 in the MVS/XA environment is that an installation can increase the size of certain critical DB2 storage pools to handle more users and tune the system to be more responsive.

In the MVS/XA environment, DB2 exploits 31-bit addressing while communicating with IMS running in a 24-bit mode. An IMS application program continues to operate in a 24-bit mode, because IMS does not support the 31-bit application call interface to DL/I or SQL.

Concluding remarks

In this paper, we have looked at DB2 from an IMS viewpoint. In the course of describing various facets of this new environment, we have identified many potential advantages that DB2 can offer to the IMS user.

One question, however, remains: How do these users decide which applications to develop using DB2 and which to develop using DL/I? For the on-line part of the application, IMS/DC facilities are used; but for the data base part some selection criteria (between DB2 and DL/I) should exist. Unfortunately the dividing line between relational and nonrelational data base applications is not very sharp. Many applications may in fact be developed using both types of data base management systems.

A considerable number of applications will continue to be best served by a nonrelational DBMS like IMS/DB. The nature of such applications is that they are using predefined and stable data base structures in a repetitive manner. For these applications, performance considerations and detailed tuning facilities may be more important than flexibility in changing and accessing data.

DB2 data bases tend to be more suited for applications where there is a clear need to use different and varying relationships in the data and where ease of change and high-level access to data is the more important aspect.

As experience with DB2 application systems grows and the implementations of such systems mature, it is likely that more and more applications will be found to be best served by relational data base systems. In time, more and more environments will discover that flexibility in access and data base change are key factors in successful data base growth and evolution.

Cited references

1. J. P. Strickland, P. P. Uhrowczik, and V. L. Watts, "IMS/VS: An evolving system," *IBM Systems Journal* **21,** No. 4, 490–510 (1982).
2. E. F. Codd, "A relational model for large shared data banks," *Communications of the ACM* **13,** No. 6, 377–387 (June 1970).
3. D. J. Haderle and R. D. Jackson, "IBM Database 2 overview," *IBM Systems Journal* **23,** No. 2, 112–125 (1984, this issue).
4. S. Kahn, "An overview of three relational data base products," *IBM Systems Journal* **23,** No. 2, 100–111 (1984, this issue).
5. C. Beeri, P. A. Bernstein, and N. Goodman, "A sophisticated introduction to database normalization theory," *Proceedings of the 4th International Conference on Very Large Data Bases,* West Berlin (September 1978), pp. 113–124; available from the Association for Computing Machinery, 1133 Avenue of the Americas, New York, NY 10036.
6. C. J. Date, *An Introduction to Database Systems* (3rd Edition), Addison-Wesley Publishing Company, Reading, MA (1981), pp. 237–272.
7. W. Kent, "A simple guide to five normal forms in relational database theory," *Communications of the ACM* **26,** No. 2, 120–125 (February 1983).
8. *IBM DATABASE 2: Application Programming Guide for IMS/VS Users,* SC26-4079, IBM Corporation; available through IBM branch offices.
9. E. F. Codd, "Extending the database relational model to capture more meaning," *ACM Transactions on Database Systems* **4,** No. 4, 397–434 (December 1979).
10. E. F. Codd, "Relational database: A practical foundation for productivity," *Communications of the ACM* **25,** No. 2, 109–117 (February 1982).
11. *Query Management Facility: General Information Manual,* GC26-4071, IBM Corporation; available through IBM branch offices.
12. *Data Extract: General Information Manual,* GC26-4070, IBM Corporation; available through IBM branch offices.

General references

C. J. Date, *An Introduction to Database Systems: Volume II,* Addison-Wesley Publishing Company, Reading, MA (1983).

C. J. Date, "Relational database: Some topics for investigation," *GUIDE 54,* Anaheim, CA, May 1982, Session MP-29.

IBM DATABASE 2: General Information Manual, GC26-4073, IBM Corporation; available through IBM branch offices.

IBM DATABASE 2: Data Base Administration Guide, SC26-4077, IBM Corporation; available through IBM branch offices.

IBM DATABASE 2: Relational Concepts, GG24-1581, IBM Corporation; available through IBM branch offices.

IBM DATABASE 2: Systems Planning and Administration Guide, SC26-4085, IBM Corporation; available through IBM branch offices.

Reprint Order No. G321-5216.

Jnan R. Dash *IBM General Products Division, Santa Teresa Laboratory, P.O. Box 50020, San Jose, California 95150.* Mr. Dash is an advisory planner in DB2 development. In 1974, he joined IBM Canada at Toronto, where he held several positions in Headquarters information systems and was responsible for designing and implementing several data base application systems. Later, he taught general data base and IMS courses to customers at the Canadian Advanced Education Center in Montreal. Outside IBM, he has worked as senior programmer analyst in Canada and data base consultant in the U.S., where he was involved in teaching IMS and designing data base applications. Since 1981, he has been part of the data systems planning organization. His current responsibility includes the DB2 product strategy. Mr. Dash received a B.Sc. in mechanical engineering from India and an M.A.Sc. in systems design from the University of Waterloo, Canada, in 1973.

Robert N. Ojala *IBM General Products Division, Santa Teresa Laboratory, P.O. Box 50020, San Jose, California 95150.* Mr. Ojala is an advisory programmer in the advanced data base group at the Santa Teresa Laboratory. He received a B.S. degree in business from the University of Minnesota in 1964. After graduation, Mr. Ojala participated in the design and implementation of many on-line systems, including message switching, order collection, engineering document control, and engineering data collection. From 1969 until he joined IBM in 1978, Mr. Ojala was an IMS systems programmer and IMS systems programming manager. He was also very active in IMS application design and in the IMS project at SHARE. At IBM, Mr. Ojala is one of the designers and implementers of the IMS attach package. He has also worked on the design of other components of DB2, including those for statistics and accounting.