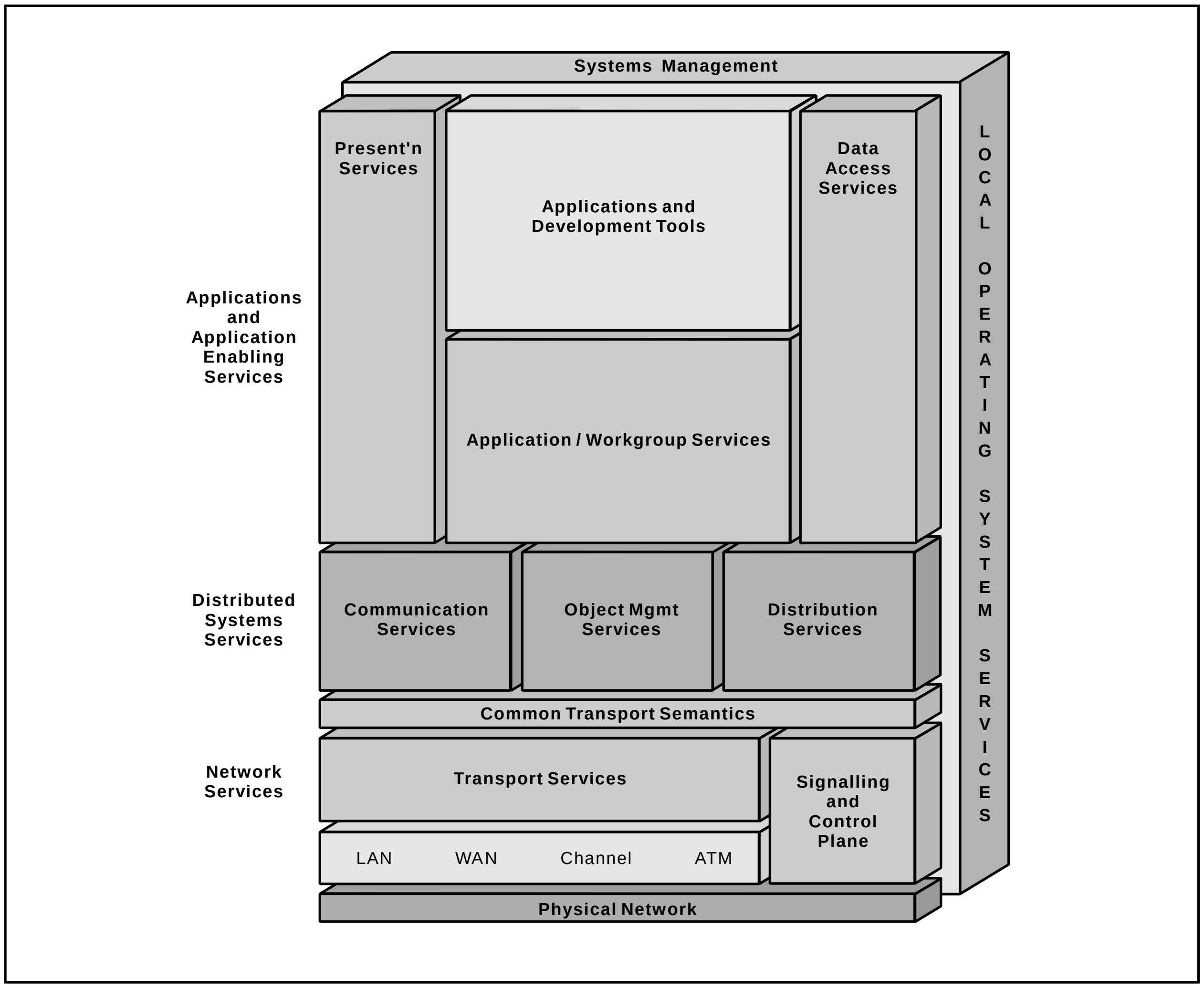
Open Blueprint

Hierarchical Database Resource Manager


Systems Management
Present'n Services
Applications and Development Tools
Data Access Services
LOCAL OPERATING SYSTEM SERVICES
Applications and Application Enabling Services
Application / Workgroup Services
Distributed Systems Services
Communication Services
Object Mgmt Services
Distribution Services
Common Transport Semantics
Network Services
Transport Services
Signalling and Control Plane
LAN
WAN
Channel
ATM
Physical Network

GC23-3918-01

Open Blueprint

Hierarchical Database Resource Manager

GC23-3918-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Hierarchical Database resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library to is provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Hierarchical Database Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Summary of Changes 1

Hierarchical Database Resource Manager 3
Application Access to Hierarchical Data 3
Logical Structure 4
Physical Structure 5
Architecture 6
Replication 7
Distributed Object Access 7
Network Computing 8
Relationship to Other Resource Manager Services 9
Solution Support 11
Characteristics 12

Notices 13
Trademarks 13

Communicating Your Comments to IBM 15

Figures

1. Hierarchical Database Physical Structure 6
2. WWW Access to Hierarchical Data 9

Summary of Changes

This revision includes:

- A description of support for making hierarchical data available to Web browser and HTTP applications
- An enhanced description of the support for making hierarchical data available to applications in the form of objects
- A description of additional relationships with other resource managers
- A discussion of solutions that can be based on the Hierarchical Database resource manager

Hierarchical Database Resource Manager

The Hierarchical Database resource manager supports applications and resource managers in IBM and non-IBM environments through standard application program interfaces (APIs). As a distributed resource manager, it also provides remote access to and interoperability among databases in an enterprise or network using the Conversational resource manager.

Application Access to Hierarchical Data

This section describes hierarchical databases and the methods that are used to access hierarchical data.

Hierarchical Databases

Hierarchical data is organized into a series of database records. Each record is composed of smaller groups of data called *segments*. Segments are made up of one or more fields. Segments relate to each other as parent and child and are defined in terms of type and occurrence of segment.

Hierarchical databases are defined to meet different application processing requirements. Data can be organized for either sequential or direct access and can be indexed. Data Entry Databases (DEDBs), for example, provide the highest levels of availability for efficient storage of and access to large volumes of detailed data.

The characteristics of a hierarchical database are defined by coding and generating a database description (DBD). A DBD describes such things as a database's organization and access method, the segments and fields in a database record, and the relationship between types of segments.

Data Language/I

Data Language/I (DL/I) is the de facto standard hierarchical data manipulation language. It provides a common high-level interface between user applications and hierarchical data. An application program reads and updates hierarchical databases by issuing calls using DL/I. Using DL/I calls, application programs can retrieve, replace, delete, and add segments to the database.

Field-level sensitivity enables an application to:

- Use a subset of the fields that make up a segment
- Use fields in a segment in a different order
- Access only selected fields in a segment for security purposes
- Add fields to a segment without affecting processing of fields that already exist in the segment

DL/I calls are invoked from application programs written in languages such as PL/I, COBOL, VS Pascal, C and ADA. The user can define data structures, relate structures to the application, load structures, and reorganize structures.

A DL/I call is made up of a call statement and a list of parameters. The parameters for a call provides:

- The function to be performed call
- The name of the data structure for the call
- The data area in the program into which the data should be returned
- A condition that the retrieved data must meet (optional)

To support distributed databases, workstation applications can have their DL/I request resolved locally or routed to a remote hierarchical database server. For example, a workstation application can transparently issue a DL/I call without knowing the data's location. The location of the data can be determined using the Hierarchical Database resource manager distribution support. The Conversational resource manager is then used with a server to obtain or update the hierarchical data.

For further information on the DL/I API, see *IMS/ESA Version 5 Application Programming: Database Manager*, SC26-8015.

Object-Oriented Access to Hierarchical Data

When application objects are to be permanently stored in some underlying data store, they are referred to as persistent objects. The services used to access persistent objects are referred to as *persistence services*, which are provided by the Persistence Services resource manager.

Persistence Services resource manager services are based on the definition developed by the Object Management Group (OMG). In addition to this definition, IBM permits different types of persistence object access. These range from a single level store access perspective, where the application references the object to materialize and store it, to access at a much more granular level that is closely related to the underlying data store. There is a class library at the granular level defined for use (if desired) when the underlying data store is a hierarchical database. The functions available in this class library directly relate to those available through DL/I.

For additional information about the Persistence Services resource manager, including how an object in a hierarchical database could be accessed using these services, refer to the *Open Blueprint Persistence Services Resource Manager* component description paper.

Additional Tools for Access to Hierarchical Data

End users want to use common tools to get data. Few have the luxury of consolidating all their data in one database manager. They want their tools to offer access to hierarchical data. Thus, access is required by other interfaces.

For example, many tools use standard relational Structured Query Language (SQL) interfaces. SQL provides access and management of data contained in relational databases and is also used for accessing data contained in hierarchical databases. It provides the capability to retrieve, insert, update, and delete data through underlying gateway products working with the Hierarchical Database resource manager. Direct SQL data access to hierarchical data from the LAN or host is provided with IBM and non-IBM gateway products. SQL calls in a client requester application can be mapped to DL/I calls in the hierarchical database server. Application calls in workstations can retrieve hierarchical data in the host this way. For example, DataJoiner provides access to hierarchical data in the host from a number of IBM and non-IBM environments. It can provide heterogeneous join (distributed request) capability across IBM and non-IBM, relational, hierarchical, and file data sources. For further information on the DataJoiner API, see the *Relational Database Resource Manager* component description paper.

Logical Structure

This section describes the logical structure of the Hierarchical Database resource manager.

Hierarchical Database Clients

The hierarchical database client provides the services required to take DL/I requests and place them on the network to get them to the appropriate hierarchical database management system (HDBMS) instance. Besides building the wire flows, the Hierarchical database client is also responsible for such things as providing appropriate security credentials with which the user can be authenticated by the server.

Hierarchical Database Servers

The Hierarchical Database resource manager server receives the request from the client and determines if it is processed locally or routed through a gateway to another instance of the HDBMS for processing. The server then packages the response from the database and returns it to the client. The server is also responsible for authenticating the user requesting database services, registering with and responding to the local Transaction Manager, notifying the Hierarchical Database client of any errors in the server and other environmental-type functions.

Physical Structure

The distributed Hierarchical Database resource manager system structure (see Figure 1 on page 6) is one that allows for distributed applications, distributed access, and replication of data. Several interfaces exist that allow flexibility and adaptability in conforming to existing networks and allow interoperability with the widest possible range of implementations.

The application uses the HDBMS-provided functional interface, DL/I. There is a direct connection between the application and the HDBMS through this interface.

The Hierarchical Database resource manager is internally structured into a client portion supporting the DL/I functional interface, a server portion, instances of which can communicate to support distributed function, and gateways that provide protocol mapping to other platforms, using the Open Blueprint Common Transport Semantics. Communication within the Hierarchical Database resource manager can occur over a variety of network transports. Typically, off the LAN, the Conversational resource manager is supported over a variety of commonly available networks. The use of gateways at the transport layer is transparent to the Hierarchical Database resource manager.

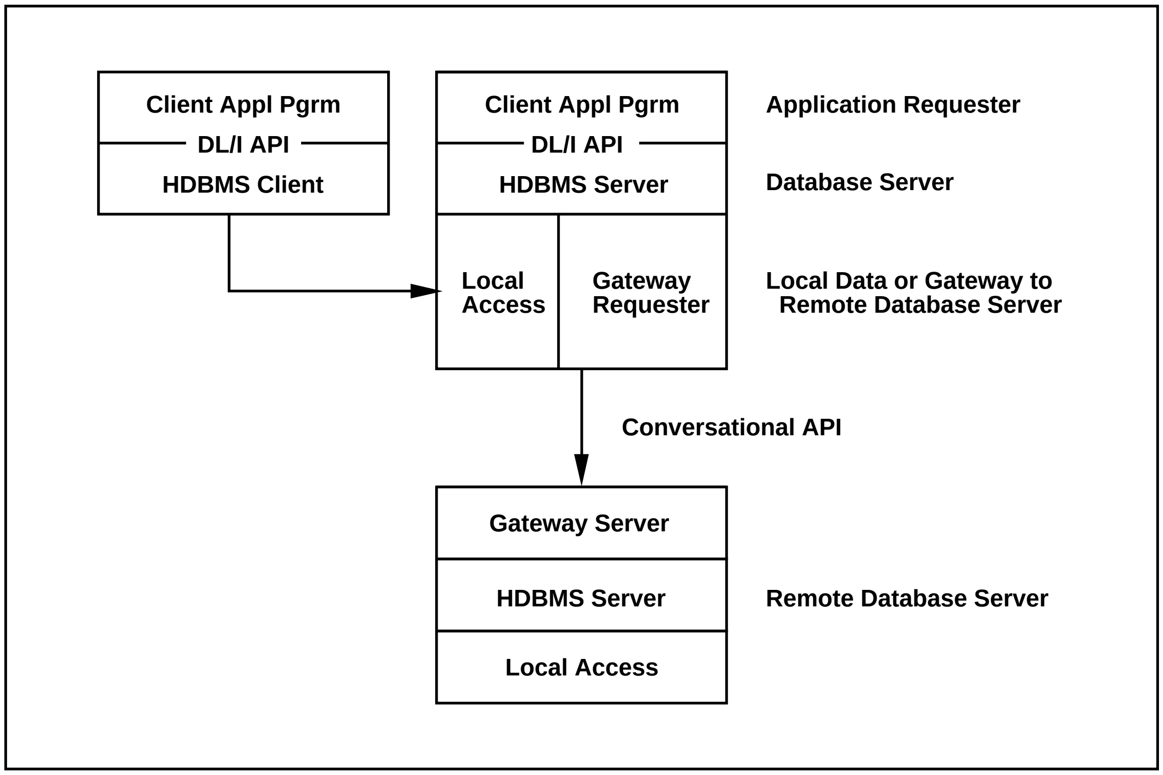


Figure 1. Hierarchical Database Physical Structure

Architecture

The Hierarchical Database resource manager distribution support architecture describes the means for distributing hierarchical data. This distribution support can be implemented in a number of different ways. This distribution support is implemented across the Information Management System (IMS) family of products by IBM and key non-IBM database and communication service providers.

The Hierarchical Database resource manager distribution support captures the call request, identifies the location of the data, triggers a requester/server pairing to pass the request, and maps the request to the source data. This requester/server pairing:

- Validates the security of the request
- Issues the call to the source database
- Transfers the data to the requester
- Includes recovery control facilities

The Hierarchical Database resource manager distribution support uses the conversational communications paradigm. Through the Common Transport Semantics, the Conversational resource manager can be used to communicate across a variety of network protocols.

One example of the use of the Hierarchical Database resource manager distribution support is MicroFocus' Remote IMS. It uses DL/I to interface to local and remote hierarchical data and uses the Conversational resource manager for communication across the distributed environment. The DBD identifies data as local or remote and initiates the conversational requester/server to retrieve the data from a remote, heterogeneous HDBMS, and provides the data to the client application. The DBD knows that the data is local or remote by using information from program control blocks (PCBs) in which parameters are added to the system definition (for example, local PCB versus remote PCB). Remote IMS also supports server-to-server communication through the Transaction Manager and the Conversational model.

Replication

The Hierarchical Database resource manager provides support for replication in conjunction with the Relational Database and Transaction Monitor resource managers, and middleware products. This replication function can extract subsets of data from a hierarchical database, and distribute and maintain copies of that data at several places in the network. By providing copies of data, performance of operations against the data can be improved by reducing or eliminating network delays and by reducing contention on the data. By automating the update of these copies, the installation provides the best data for an application at the right place in the network.

Creating copies of data also assists an installation in maintaining separation between operational use of data, which can involve updates, and informational or decision support use of data, which could require read-only capability. This ensures that mission-critical operations are not impacted by more resource-intensive informational querying. It also ensures that locks held by updaters (exclusive locks on a small amount of data) do not prevent informational querying across large amounts of data.

Replication functions include automatic facilities that maintain up-to-date copies (to the specified level of currency) without intervention by operators or users. These facilities can implement this activity outside of prime time with differential for the local time zones and peak loads. This is accomplished by using asynchronous distribution facilities that separate the creation of the updates, capture, transfer, and apply the updates to the replicated database system. The capturing system creates a file of changes and distributes the file asynchronously to the database manager where the copies reside. There can be several capturing and target sites. Each site can process the update at its own convenience.

There are a number of techniques that assist with replication. One technique is to replicate the hierarchical data into a relational database. This approach uses initial data transfer and continual update to maintain a relational instance of hierarchical data and to extend that to the workstation environment. The initial data transfer extracts hierarchical data out and puts it into a relational data base, so that SQL can query the relational database.

Continual update, with the data capture facilities of the database manager, captures relational or hierarchical data as changes occur, maps that data to a corresponding database, and updates the corresponding database. The databases can reside on the same or different systems. The update frequency can be set according to the business requirements.

One example of the use of the Hierarchical Database resource manager support for the replication function is through the IBM Data Refresher and Data Propagator products. With these products, data can be extracted from IBM's IMS Database Manager (IMS DB) databases and placed into relational databases, either locally or remotely. Updates can be propagated to the relational database or propagated back from the relational database to the hierarchical IMS DB database.

Data captured in this manner can be used as history data in trend analysis applications. These capabilities should extend the range of relational tools and applications to hierarchical data. They can augment existing host data with local workstation based processing.

Distributed Object Access

The Hierarchical Database resource manager provides support for distributed object access in conjunction with the Object Management Services, and the Transaction Monitor and Persistence Services resource managers. This distributed object access function can extract subsets of data from a hierarchical database as objects, distribute and maintain copies of that data at several places in the network, and send updates back to the hierarchical database when needed.

IMS provides a three-tier, client-server-host environment for workstation object access to hierarchical data. The IMS Client Server Object Manager (IMS CSobject) provides an object-oriented interface to a client application. The client application can use the IMS CSobject to manipulate objects. Hierarchical data is repackaged as objects, placed in a persistent cache, and managed on a local server. Connection to hierarchical data from IMS CSobject is through communications services and transaction monitor application programs that access the hierarchical datastore and communicate with callable services. Requests for data are routed in this manner from the server to the hierarchical datastore on the host.

With IMS CSobject, new data objects can be created that correspond to existing IMS DB data while meeting the requirements of new object-oriented application programs. Host data can be redefined as objects that represent entities within a real-world system. Data and function can be repackaged for easier reuse and maintainability without impacting the investment in current databases. Actions can be associated with each object, creating an independent unit of data and function that client applications can use and reuse. Each object can correspond directly to an IMS record, or the object can be a completely new structure that combines parts of different data structures in different databases -- whichever the client application program requires.

Objects created are based on IBM's Systems Object Model (SOM) and inherit the SOM class structure. The extensive facilities provided by SOM and those provided by IMS CSobject can be used to develop new client application programs. When a client application program requests an object, it does so through the IMS CSobject support of the Hierarchical Database resource manager. IMS CSobject retrieves the hierarchical data, converts it into object form, and stores it as an object on an intermediate server, thus creating an object instance of the hierarchical data, and maintains the local cache. The object remains on the server unless it is requested that the hierarchical database be updated with the changes contained in the data object. While the data is on the server, client application programs can interact with the data independently of the hierarchical database on the host.

Client application programs can be written that add or update data objects without referring to their locations. To do this, IMS CSobject uses the Object Request Broker resource manager facilities and TCP/IP, NetBIOS, or NetWare IPX/SPX to route the request to the server that contains the object.

Network Computing

World Wide Web (WWW) access to hierarchical data is available. This capability uses the facilities of the Hypertext Transfer Protocol (HTTP) resource manager's Common Gateway Interface support (HTTP CGI) for mapping data requests between the Web Browser and Transaction Monitor resource managers. Transaction Monitor resource manager applications act on the mapped input using the Hierarchical Database resource manager in the normal fashion and supplying results to the requester using the CGI. The application can use Open Blueprint Communications Services for distributed access to the data. Figure 2 on page 9 shows these relationships and flows.

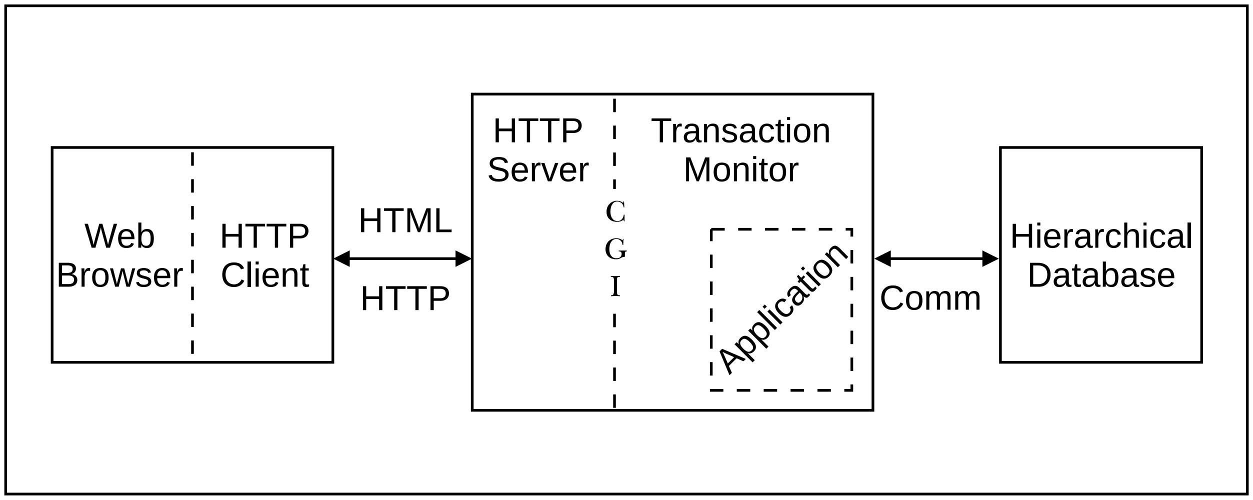


Figure 2. WWW Access to Hierarchical Data

Relationship to Other Resource Manager Services

This section describes the relationships between the Hierarchical Database resource manager and other resource managers in IBM's Open Blueprint.

Communication Services

The Hierarchical Database resource manager uses the conversational paradigm for access across the network for distributed DL/I work. Interoperability across any of the transport protocols can be supported through Common Transport Semantics and gateways without requiring all protocols to be supported on all platforms or requiring an implementation to support all protocols.

CGI, which is part of the HTTP resource manager, allows users Internet access to hierarchical data. HTTP gateways are described that can be used for hierarchical data access.

Object Management Services

The IMS Client Server Object Manager of the Hierarchical Database resource manager uses object management services for communications.

Distribution Services

Identification and Authentication Services: The Hierarchical Database resource manager depends on the underlying support of the Conversational resource manager to honor the user registration and logon provided by the Identification and Authentication resource manager. The Hierarchical Database resource manager does not require an additional user registration process or a separate end user authentication dialogue.

Directory Services: The Hierarchical Database resource manager depends on the directory usage by the Conversational resource manager. The client requester can then obtain the hierarchical database LU name and other binding information from the Directory resource manager to access the hierarchical database through the Conversational resource manager.

Each HDBMS instance knows where the databases are located based on system definitions (for example, database definition, program specification block generation, and so on) and dynamically allocates resources as needed.

Transaction Management: HDBMSs provide services to create, destroy, insert, update, and delete data that operates in the context of a logical unit of work (LUW). This work is performed within a scope that is terminated by commit or backout calls. When commit is performed, all the HDBMS work requested by the application is “hardened” in the database. When backout is performed, all tentative changes to the database are deleted.

The HDBMS depends on the Transaction Manager resource manager to support a broader unit of work where more than database resources are involved, in which case the HDBMS would respond to calls from the Transaction Manager resource manager. The original request for commit (or backout) would initiate from the application using the Transaction Manager API. During syncpoint processing, the Transaction Manager resource manager drives the Hierarchical Database resource manager to do the necessary two-phase commit processing. The Transaction Manager resource manager acts as the coordinator and the Hierarchical Database resource manager acts as the participant.

For local users, the HDBMS associates a logical unit of work identifier (LUWID) with the user's work as soon as the user makes a request that requires a LUWID. It is assumed that the user might have already done some recoverable work before execution of the database request began. The HDBMS obtains the current LUWID from existing system information.

Application Services

Transaction Monitor: Applications can use the Transaction Monitor resource manager to access hierarchical data. This resource manager provides applications with access to a variety of application and system services and addresses aspects of program execution, security, system management, and transactional and service integrity for the hierarchical data.

The HTTP resource manager CGI support enables Internet access to hierarchical data through the facilities of the Transaction Monitor resource manager.

Collaboration Services: Applications can use the Collaboration resource manager to access hierarchical data. Lotus Notes handles document management and other forms of collaboration, both asynchronous and real time (including audio-video conferencing). Collaboration resource manager access to hierarchical data is provided through the Messaging and Queuing resource manager.

Data Access Services

Persistence Services: Object-oriented applications can use the Persistence Services resource manager to access hierarchical data. For additional information about the Persistence Services resource manager, including information about how an object in a hierarchical database can be accessed using these services, refer to the *Open Blueprint Persistence Services Resource Manager* component description paper.

Application Development

Application development tools help the application developer implement distributed hierarchical applications that use standard interfaces and the facilities of the Open Blueprint. The application development tools assist the database administrator in defining hierarchical database definitions. They assist the application programmer in providing language support for the DL/I call interface through segment search arguments (SSAs). Application development (AD) support provides the tools and languages that together with the class libraries of persistence services, enable object-oriented applications to use the Hierarchical Database resource manager to store and retrieve their data objects. Additional AD tools can further simplify the creation of workstation applications for access to hierarchical data.

Systems Management

The Systems Management components of the Open Blueprint structure provide the capabilities to manage resources across open environments in a seamless fashion with minimum human intervention and maximum efficiency. The Hierarchical Database resource manager provides functional and management support of its resources. Through the existing APIs, systems management applications access the Hierarchical Database resource manager systems management support functions. These support functions use the Open Blueprint Systems Management services as an important portion of the Hierarchical Database resource manager systems management support.

Local Operating System Services

The distributed Hierarchical Database resource manager utilizes the local system services that support the Open Blueprint structure to manage local system resources such as memory, CPUs, and devices.

The language environment support provides standardized services (for example, storage allocation and management) to applications accessing hierarchical data through a standard set of calls. These facilities are based on POSIX initiatives for common operating system services and interfaces to those services, such that portability of the components and applications using those interfaces and services is possible and can be facilitated. For example, the MVS HDB RM, IMS/ESA DB, supports the POSIX standard today through the language environment support and the local operating system services provided by the MVS OpenEdition product, enabling portability for its components and associated applications.

Solution Support

This section describes solutions that use the Hierarchical Database resource manager.

Information Warehouse

The Hierarchical Database resource manager provides mission-critical data for Information Warehouse solutions. This is enabled by data access and replication facilities. Administration and systems management capabilities are used to control the flow of hierarchical data into the Information Warehouse, where it is combined with other data (such as relational) that is provided in the Information Warehouse. Data presentation can be enhanced by using graphical user interfaces and multimedia. New technologies can be employed, such as audio and video data, touch-screen user interfaces, and the like, to create new client application programs for access to hierarchical data that meet the requirements of users in a user-friendly, interactive manner.

Data Mining

The Hierarchical Database resource manager provides mission-critical data for data mining. This is enabled, for example, with the facilities of the Intelligent Miner, which provide data extraction, formatting, decision support, and reporting. These facilities are used with hierarchical data and with other data (such as relational) that is provided for use in data mining.

Characteristics

Characteristics of an open, heterogeneous, distributed system are provided by the distributed Hierarchical Database resource manager. Appropriate performance, reliability, flexibility and integrity are provided for individual environments and for the distributed system as a whole. Hierarchical data is made easily accessible with transparent access. The Hierarchical Database Management System and the distributed environment in which it runs are easily manageable and able to accommodate a wide variety of environments.

Scalability

The distributed Hierarchical Database resource manager can address an essentially unlimited scale. It can scale down to smaller networks while maintaining appropriate performance and costs. A few workstations and a server system on a LAN can constitute a legitimate hierarchical distributed system. In larger networks, each individual workstation can be relatively small due to overall cost considerations. The distributed hierarchical structure permits the subsetting of functions for smaller machines. The hierarchical distributed structure can also be implemented on large networks and can use the largest individual processors or parallel sysplex to provide the highest performance and most cost-effective solution. The hierarchical distributed structure can accommodate incremental HDBMS servers as needed for scalability. Access is provided, for example, through Remote IMS from any server to any number of additional servers. The distributed hierarchical system can accommodate a choice of configurations from relatively low cost and few functions to many functions with high reliability and performance.

Transparency, Portability and Interoperability

The Hierarchical Database resource manager is currently supported on mainframe and workstation platforms (OS/2, Windows, AIX, HP-UX, SUN Solaris, MVS, VM, and VSE) across a number of IBM and non-IBM products. The Hierarchical Database resource manager provides support for the standard DL/I API and its definitions in these environments. It allows transparent access to hierarchical data from each environment and portability of applications accessing hierarchical data from each environment as well. Distributed hierarchical support allows transparent access of hierarchical data between a client requester to either LAN or host servers, or both.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
IBM
IBMLink
IMS
IMS/ESA
Information Warehouse
Language Environment
MVS
Open Blueprint
OpenEdition
OS/2
SOM
System Object Model

The following terms are trademarks of other companies:

HP	Hewlett-Packard Company
IPX	Novell, Incorporated
Lotus Notes	Lotus Development Corporation
POSIX	Institute of Electrical and Electronic Engineers
Solaris	Sun Microsystems, Incorporated

Microsoft, Windows and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3918-01