MARCH 1983 VOL. 6 NO. 1

a quarterly bulletin of the IEEE computer society technical committee on

Database Engineering

Contents

Letter from the Editor 1

Letters from the Guest Editor 2

Some Characteristics of the Japanese Computer Industry 3
Y. Kambayashi

An Enhanced Relational Data Base System for Planning and Management Information Processing (PLANNER) 5
A. Makinouchi, M. Tezuka, and Y. Kanda

AIM: Advanced Information Manager 9
M. Minami

Extended Data Management System (EDMS) 13
K. Taguchi

Online Database Management System 17
K. Masamoto

Relational Data Base Systems INQ and RIQS 22
K. Hara, T. Gotoh, T. Miyazaki, K. Takeuchi, and S. Mabuchi

Introducing ADABAS to the Japanese Market 29
Y. Ishii

Overview of IMS/VS Fast Path Enhancement .. 33
T. Takeshita

Database Management Systems for Very Large Scale Applications 40
K. Suzuki and S. Ikeda

The Data Communication System for Nationwide Banking System 45
H. Imamura

CADETT: Computer Aided Design and Engineering Tool for Toyota 48
Y. Sakai and Y. Kuranaga

The Travel Reservation On-Line Network System 53
K. Tsukigi and Y. Hasegawa

Heterogeneous Distributed Database System: JDDBS 58
M. Takizawa

Database Machine Activities in Japan 63
S. Uemura

Database Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Database Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Database Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in the Database Engineering Technical Committee is open to individuals who demonstrate willingness to actively participate in the various activities of the TC. A member of the IEEE Computer Society may join the TC as a full member. A non-member of the Computer Society may join as a participating member, with approval from at least one officer of the TC. Both a full member and a participating member of the TC is entitled to receive the quarterly bulletin of the TC free of charge, until further notice.

Membership applications and requests for back issues should be sent to IEEE Computer Society, P.O. Box 639, Silver Spring, MD 20901. Papers and comments on the technical contents of Database Engineering should be directed to any of the editors.

Letter from the Editor

This issue is devoted mainly to reporting the current status of commercial database and database application systems developed in Japan or developed elsewhere but enhanced in Japan to meet the special requirements imposed by the Japanese market. Since database journals and conferences in the United States and Europe are not exactly inundated with papers from Japan, we felt that the topic may be of substantial interest to the database community outside of Japan. Prof. Yahiko Kambayashi of the Kyoto University, Kyoto, Japan, and I have jointly put this issue together. In view of the communication and language difficulties that the Pacific Ocean presented, I was very fortunate that Prof. Kambayashi served as Guest Editor to edit this issue with me. It was he who drew up the list of corporations from whom to solicit papers. Although for a number of reasons some of them were initially reluctant, he prevailed upon all of them to submit papers. He had to also perform the unpleasant task of dunning the authors (in a society where dunning is particularly unpleasant) to meet the submission deadlines for the initial, revised, and final versions of their papers.

I would like also to extend my special thanks to Dr. Yoshifumi Masunaga, who is visiting IBM Research, San Jose, from Tohoku University, Sendai, Japan. He played an important role in shaping the editorial direction of this issue by providing me with helpful comments on several of the submitted papers and educating me on the database activities in Japan.

As in the past, we will continue to devote each issue to one area of research relevant to engineering aspects of database architecture. We have chosen the following topics for the next four issues.

1. Highly Available DBMS: Dr. Dan Ries is putting together the June issue on this topic.

2. Expert Systems. Dr. Adrian Walker (IBM Research, San Jose) has agreed to serve as Guest Editor for the September issue on this topic. Submission deadline is May 1.

3. Automated Office Systems. Prof. Fred Lochovsky, who is visiting IBM Research, San Jose, from the University of Toronto, will serve as Guest Editor on this topic for the December issue. Submission deadline is August 1.

4. Statistical Database Management. Prof. Don Batory will be in charge of the March 84 issue on this topic. Submission deadline is November 1.

Although we expect that each issue will consist mainly of papers that we will invite from leading research/development groups in the particular topic chosen, we will accept up to a few papers that are not related to the special topic and include them in a separate section in any issue. Papers that are relevant to the special topics should be submitted to the editor in charge of the issue and to me (one copy to each). All other papers should be submitted to me.

Won Kim

Letter from the Guest Editor

Most of the articles contained in this special issue of Database Engineering have never appeared in English before. I believe that we Japanese should be much more active in publishing our research and development activities in English in order to promote closer working relationships with our American and European colleagues. The number of papers by Japanese authors that have appeared in English-language journals and conferences has been a small fraction of papers that are published in Japanese journals. I hope this issue will serve to trigger a more active participation by Japanese authors in the development and exchange of ideas with our colleagues outside Japan. I am grateful to Dr. Won Kim for providing us with an opportunity to report on our database activities.

Although Japan's hardware technology is quite advanced, its software technology has considerable room for improvement. Until recently, database systems have not been widely used in Japan. It was mainly because of the high cost of processing the Japanese language. This problem has largely been resolved now, and database systems are being recognized as essential for improved productivity. Articles contained in this special issue have been selected to acquaint the American and European readers with a broad spectrum of database products developed in Japan, the special requirements imposed by the Japanese market, and some database application and transaction-processing systems currently used in Japan.

I would like to take this opportunity to thank the authors who contributed articles to this issue. They spent a lot of time preparing, revising, and re-revising the papers in English. Because of the English problem that many of the authors had, Dr. Kim had to work especially hard. He extensively commented and corrected the first versions of all the papers and actually completely rewrote several of the papers. His efforts considerably improved the technical contents and readability of the papers that appear in this issue. Without his enthusiasm and hard work, this issue would not have materialized. I enjoyed assisting Dr. Kim with the selection of papers and preparation of the issue.

I would like to also thank Dr. Hirotaka Sakai, current chairman of SIGDB of the Information Processing Society of Japan, for his comments on the selection of contributors, and Dr. Yoshifumi Masunaga, who visited IBM San Jose Research during 1982, for assisting me with communications with Dr. Won Kim.

I hope the readers will find this issue interesting and informative.

Yahiko Kambayashi

Some Characteristics of the Japanese Computer Industry

Yahiko Kambayashi

Department of Information Science
Kyoto University
Sakyo, Kyoto, Japan

Two major characteristics distinguish the Japanese industry from the American industry. One is that the Japanese industry consists of several company groups, each of which includes various companies providing a variety of services and products, such as banks, trading companies, and computer companies. The other is the life-time employment. Although the bond among the companies within a company group has been weakening and life-time employment is no longer being accepted as a fact of life, these two conditions characterize the Japanese computer industry as well. The tendency for each company group to want its own computer company has contributed to the establishment of four major companies that manufacture large mainframes: Fujitsu, Hitachi, Nippon Electric Corporation (NEC), and Mitsubishi. Fujitsu is currently the top computer company in Japan, and Mitsubishi is the smallest of the Big Four. All four companies belong to different company groups. All of them sell mainframes, office computers, super minicomputers, personal computers as well as ICs like the 64K memory chips. Fujitsu and Hitachi manufacture Cray-like supercomputers and IBM-compatible mainframes.

Although it seems to be a widely held belief in the United States that the success of the Japanese computer industry is largely due to the control and support from the Japanese government, this author believes that competition among the companies is the major reason. Also because of the life-time employment, worker resistance to the introduction of computers and robots has been minimal, compared with the American industry. This explains why the the computer market in Japan is so large; the number of computers installed in Japan is second in the world.

Since the Constitution prohibits militarization of Japan, research support from the military department is almost nonexistent. Most Japanese computer companies receive support from Nippon Telephone and Telegraph Corporation (NTT) and the Ministry of International Trade and Industries (MITI) through their research and development projects.

NTT, which used to be a governmental agency, became a public corporation over 30 years ago. They have their own research laboratories and developed their own computers, called DIPS-11 series (Models 5, 15, 25, 35, 45, and VLSI version) for electronic switching and computer-center services. Fujitsu, NEC, Hitachi and Oki produce various communications equipments, and purchases from NTT are very important to these companies. Having both the computer and communications technologies is indeed advantageous for Japanese computer companies. NTT also sponsors projects to develop new systems. The projects usually involve more than one computer companies. The INS (Information Network System) project it is currently undertaking is very influential for the future of computer applications in Japan.

In the past, MITI sponsored projects like VLSI development and pattern information processing. Currently, it supports the fifth-generation computer project through a newly formed research institute called ICOT (Institute for New Generation Computer Technology). It is the first long-range research-oriented project and its scope includes database machines and knowledge-based systems. The project is open to foreign researchers and companies. The main research laboratory of MITI is the Electro-Technical Laboratory, which is located in Tsukuba, a new city consisting of university and research centers. MITI also financially supports the Japanese

Information Processing Development Center (JIPDEC), an educational organization for the Japanese computer industry.

Several years ago, in order to position the Japanese computer industry to effectively compete with IBM, MITI ordered the formation of three computer-company groups: Fujitsu-Hitachi, NEC-Toshiba, and Mitsubishi-Oki. These groups are no longer functional. The first group announced the M-series computers, M-160, 170, 180, 190, and 200, that are IBM-compatible. Since then, however, Fujitsu developed the M-3xx series and Hitachi the M-2xx, which are not compatible with each other. The NEC-Toshiba group has been dissolved, since Toshiba decided not to make mainframes. In June 1982, Mitsubishi and Sperry-Univac entered into a joint venture agreement. Since Oki is the parent company of Oki-Univac which produces small and medium-scale computers for Univac-Japan, the Mitsubishi-Oki group appears to be working with Univac.

Two important requirements characterize the Japanese computer market. One is the Japanese language processing capability. The other is that users tend to require their own customized software packages. Until recently, the Japanese have had to rely on the English alphabet and the 46 Japanese phonetic symbols called Kana to express Japanese sentences. One of the major reasons that have impeded a wider use of database systems in Japan has been the inadequate Japanese language processing which includes over 3,000 Kanji (Chinese) characters. This problem has been resolved to a good extent. The reason why users require customized software products is mainly that most Japanese employees stay in one company for life and thus the way of doing similar jobs is not standardized across companies.

Foreign computer companies, with the exception of IBM, have had problems establishing themselves in Japan. User surveys indicate three reasons. First, they have weak sales networks. Second, the Japanese language processing capabilities in their systems are in general not adequate. And third, they usually refuse to provide customized software packages and instead insist on selling foreign-made general-purpose packages. Further, because of the substantial discounts on Japanese computers sold to educational institutes, most computers installed in Japanese universities are Japanese made.

The following shows the shares of the computer market in Japan that domestic and foreign computer companies have for computers that cost not less than 30 million yen (1$ = 230 yen). (The data are taken from the January 1983 issue of Computopia.)

market share by sales

IBM (27.6%) Fujitsu (21.1%) Hitachi (16.6%) NEC (14.1%)
Univac (10.4%) Burroughs (4.1%) Mitsubishi (3.0%) NCR (2.0%)
others (1.1%)

market share by the number of computers installed

Fujitsu (29.0%) NEC (20.8%) Hitachi (15.7%) IBM (11.4%)
Mitsubishi (7.3%) Univac (6.6%) Burroughs (4.9%) NCR (3.4%)
others (0.8%)

IBM is very strong in the area of large computers, with 49.2% of computers costing not less than 1 billion yen.

Acknowledgement: Dr. W. Kim rewrote an earlier longer version of this article by extracting contents which appear to be of interest to the American readers. This article only reflects the author's personal view.

An Enhanced Relational Data base System for Planning and Management Information Processing (PLANNER)

Akifumi Makinouchi*, Masayoshi Tezuka*, Yasunori Kanda**

1. Introduction

PLANNER (Planning and management information system based on Easy Relational data base) is a total system for planning and management. The System runs on FACOM M-series main frames under OSIV/F4 or OSIV/X8 since late 1981.

PLANNER incorporates a flexible data base system in order to allow end user to efficiently retrieve data for routine processing, forecasting and planning. The data base system can handle diverse and voluminous information and has facilities for supporting a powerful end user language that allows efficient data base inquiry. It also provides facilities for processing, editing, and analyzing data collected by end users in a trial-and-error fashion, as well as capabilities for graphic display and mapping facilities to show related geographical data.

2. The PLANNER System

The configuration of the PLANNER system is shown in Fig. 1. The system consists of a command interface, a relational subsystem, and application subsystems. The command interface distributes user commands to the relational subsystem or to an appropriate application subsystem, according to whether it is a query or an application command.

PLANNER is an open-ended system so that an application subsystem may be integrated into it by using the ISRQ (Interactive Subsystem for Relational Query) interface and/or the DSCS (Data Storage and Control Subsystem) interface. In addition, the interactive PLANNER provides a facility that allows an application subsystem to define and access application-specific data attributes through the data dictionary shared with the relational subsystem.

---

* Software Laboratory, FUJITSU LABORATORIES, LTD.
1015 Kamiodanaka, Nakahara-ku, Kawasaki, JAPAN 044-777-1111

** Management Systems Development Dept., FUJITSU LTD.
1-17-25 Shinkamata, Ohta-ku, Tokyo JAPAN 03-735-1111

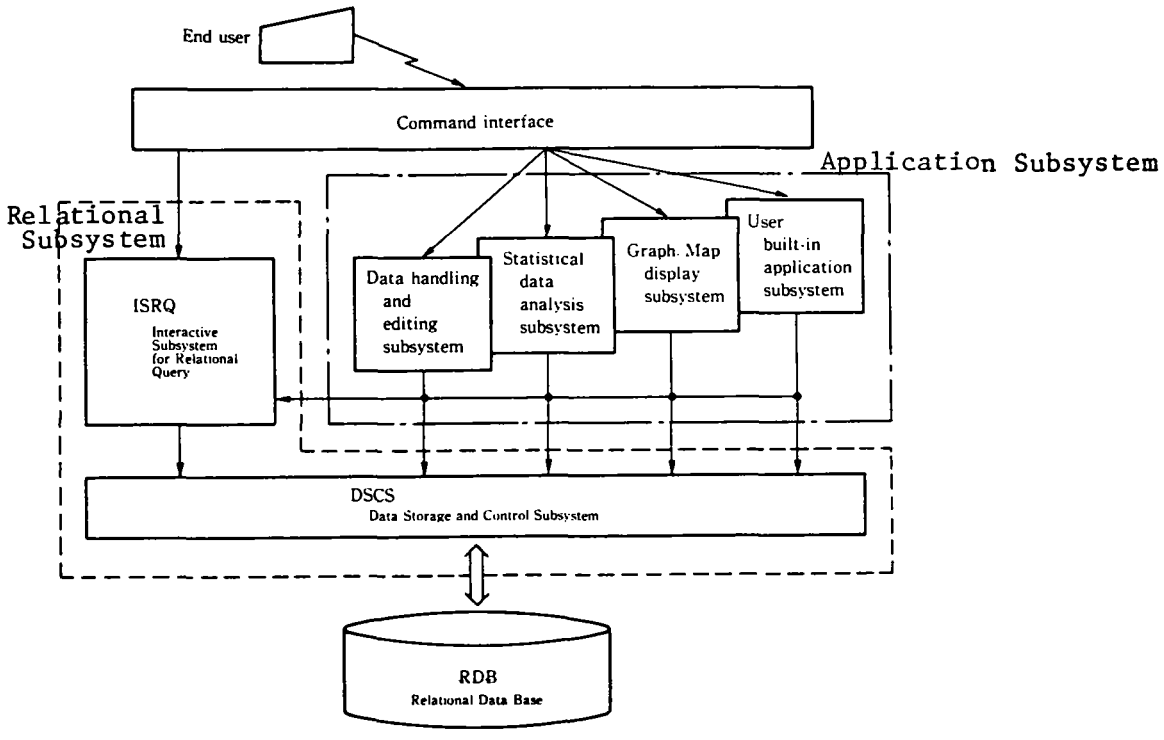


Fig. 1 PLANNER system configuration.

The relational subsystem consists of ISRQ and DSCS. ISRQ[1)] interprets and executes the query language for the relational data base. The query language contains commands to define or delete tables, indexes, and views, as well as commands to retrieve, display, insert, delete, or update records.

DSCS supports both data organization and access methods. There are three types of data organization; entry sequential, index, and hash.

An entry sequential table contains records stored in the order of their arrival. Index and hash tables have a B*-tree structure, and, (key value, record value) pairs are stored in the leaf pages of the B*-tree.

In PLANNER, indexes can be defined as access paths for the entry sequential tables. Indexes are implemented in the index table supported by DSCS. The DSCS interface provides facilities for table definitions, for scanning and fetching tables, and for inserting, deleting, and updating records. DSCS provides I/O management, a lock mechanism to support multiple concurrent users, and a log facility for data base recovery.

3. Unified Language for End Users

The language supported by PLANNER is a unified one in that it allows end users to interrogate a relational data base and to analyze the retrieved data. This language is called QAL (Query and Analysis language).

Although QAL is a combination of two command languages for end users, it is syntactically and semantically unified so that users perceive it as a single, integrated language.

One part of QAL, called RQL, is also a unified query language for relational data bases, with facilities to define or delete tables, to retrieve, insert, or update records, and to grant access rights.

The other part of the language, AL, has many features which allow users to analyze retrieved data, edit the results, and output them to a video display terminal or a printer in graphic form. There are 25 commands in RQL and about 90 commands in AL.

3.1 RQL

RQL is syntactically and semantically similar to the SEQUEL II and QUEL languages[2),3)] with minor differences. RQL commands can be easily used by non-professional users. Records retrieved from one or more tables using the RQL GET command are places in a logical work area called temporary segment. RQL provides data manipulation commands such as INSERT, DELETE, UPDATE as well as set-operations such as intersection, difference, union, and append.
Views are retrieved like basic tables, but can not be updated.

3.2 AL

AL provides various facilities to allow users to handle specific data in planning and management information. Some of the commands supported in AL are explained here.

Commands for analyzing time-series data and for extended data handling

The COMPTS command calculates the difference or ratio of periodically collected data. For example, it can compare data from one quarter of last year with the data from the same quarter of this year. The TRTS command converts one kind of time-series data (e.g. monthly data) to another kind (e.g. yearly data). The GROUP BY function and the join operation have been extended for the RANK and MERGE commands, respectively. RANK allows users to group records by value ranges in a specified field of a table. MERGE is an extension of the join operation, similar to the outer join operation.

AL also have array handling capabilities. The EXTRACT command generates arrays from a table and the BIND command cenverts arrays to a table form.

Statistical analysis commands

AL statistical commands include STAT, CORR, CROSSTB and FACTOR. The STAT command calculates fundamental statistical values such as maxima, minima, summations, averages, counts, and standard deviations. The CORR command is used to calculate correlation coefficient. The CROSSTB command classifies records in a table according to value ranges on specified fields, counts the number of records in each class, and displays them in table format.

Commands for displaying outputs in graphs and maps

Users can enter commands to display the results of analysis in graphs or maps. Facilities are available for displaying many types of graphs (bar graphs, line graphs, pie charts, and histograms), especially suitable for analyses of area or time-series data. Map commands are divided into two general classes: one for grid maps and the other for district maps. A grid map command refers to grid tables and classifies grids according to data associated with the grid (e.g. population in the grid) and plots a symbol for each grid on the map. A district map command referes to tables in which the key field is a district code (e.g., the population density of an area). On the district map, districts are hatched differently to classify districts.

Acknowledgement

We wish to express our sincere thanks to Mr. Hayashi of FUJITSU LABORATORIES LTD. and to Mr. Sunada of FUJITSU LTD. for their contribution.

We wish to express our sincere thanks to Dr. Kim for editing this PLANNER paper.

References

1) A. Makinouchi et al.: "The Optimization Strategy for Query Evaluation in RDB/V1," Proc. 7th Int. Conf. VLDB, pp.518-29 (Sept. 1981)

2) D.D. Chamberlin et al.: "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM J. Res. Develop., pp.560-75 (Nov. 1976)

3) M. Stonebraker et al.: "The Design and Implementation of INGRES," ACM TODS, 1,2, pp.189-222 (Sept. 1976)

AIM: Advanced Information Manager

Masahiro Minami
Software Division
Fujitsu Limited
1015, Kamikodanaka
Kakahara-Ku, Kawasaki, Japan

1. Introduction

In recent years, the amount and variety of data processing has required more powerful database and data communication(DB/DC) systems. The most important goals of DB/DC systems are high efficiency, high reliability and ease of use and maintenance. AIM, the DB/DC system for the Fujitsu FACOM M series computer, was developed to meet these requirements. To achieve high efficiency AIM was designed as an integrated database and data communication system to avoid the redundant overhead, the duplication of functions between the DB and DC system and also between AIM and the operating system. AIM supports multitasking, very fast access method dedicated to databases, optimal buffer management, and resident database in virtual memory. For high reliability it provides recovery/restart, dual file support for both the log and database, and shared database management by multiple systems. Further, it provides functions to improve productivity of design development, management and maintenance of application systems, such as database design and application program development tools based on the dictionary and debugging/testing tools.
In this paper, an overview of the system will be presented.

2. System Configuration

AIM consists of four major components as shown in Fig 1. The DCMS(Data Communication Management sub-System) is the component which is responsible for handling messages between terminals and application programs. DCMS supports message distribution, message editing and message communication between application programs. Thus, an online application system can be easily implemented for small-scale systems and very large-scale systems.

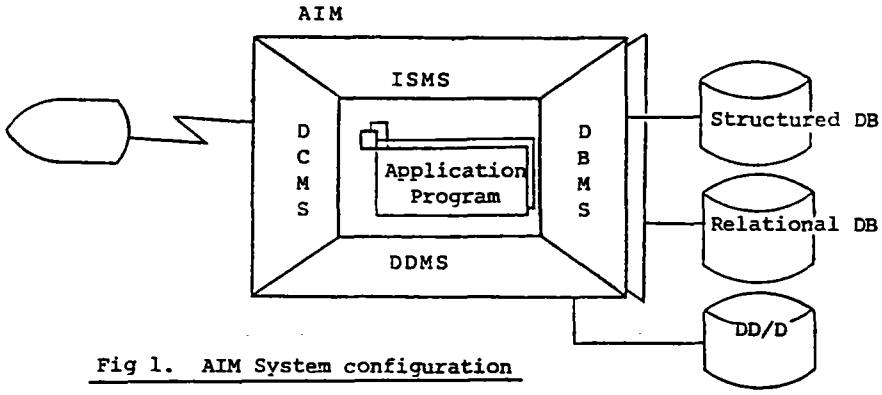


Fig 1. AIM System configuration

The ISMS(Integrity and Schedule Management sub-System) is the component which supports integrity and reliability. It provides such functions as automatic log collection, access control, recovery and restart and the command scheduler during online database processing.

The DDMS(Dictionary and Directory Management sub-System) is the component which manages all information related to AIM such as the data structures, terminals, message structures, application programs and resources for recovery/restart. This information is sotred in DD/D. It has a definition language called ADL(AIM Description Language) for this information.
DBMS(DataBase Management sub-System) is the component which provides the various access facilities for the database. AIM has two DBMSs; one for handling structured databases based on the CODASYL data model 1), and the other for relational databases. The former is handled by DML(Data Manipulation Language)2), and the latter by AQL (Advanced Query Language) wihch is similar to SQL.3) Both languages may be used in the same application program, and both methods are managed by the same functions for data integrity. These DBMSs are selected on the basis of the nature of the data or the data processing requirements.
In this paper, the facilities of the DBMS for only the structured database will be described.

3. Database Structure

A database is structured on four levels as shown in Fig 2. The logical level can be expressed in the network form by combination of RECORDs and SETs.

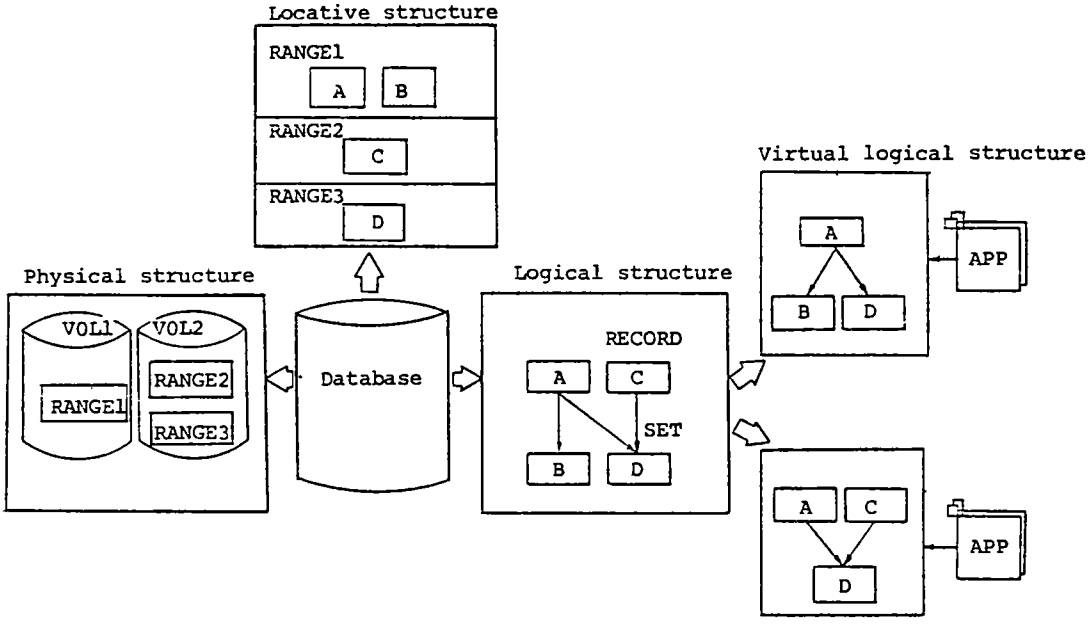


Fig 2. Data structure

The locative structure consists of the RANGEs in which the RECORD(s) are distributed. The user can define locative information, such as the arrangements of RECORD(s), the storage in a RANGE and its overflow pool and the blocking ratio(e.g. according to access frequency) without being aware of the physical DASD attributes. The physical structure consists of the physical DASD volume(s) where the data are stored.
The virtual logical structures are composed from the logical structure.
These logical, locative and physical strucutres are stored in the DD/D as the SCHEMA and the virtual logical structures are stored in the DD/D as the SUB-SCHEMA. These structures are managed by DDMS so that independence among levels can be established.

4. Relationships between records

The relationships between records can be established by means of the stored address (pointer) in the control part of the records or by using(the key of) a common data item between different record types. The former is called a physical concatenation, and the latter a logical concatenation.
Two forms are used for the physical concatenation which indicates the hierarchical relationship between the owner record and the member records, RING and LIST forms. In the RING form, one or more member records are connected to an owner record in a loop. In the LIST form, the owner record contains all of the pointers of the member records related to the owner record. The LIST form is effective in managing time-series data. The system supports the multidimensional(pointer array) management for the LIST form.
The logical concatenation is a means of relating records belonging to different record types according to the key values of the records. Logical concatenation is not concerned with the actual addressed of the data, so reorganization of records is simplified.

5. Access method enhancements

AIM provides two basic access methods. First is the direct access method which is data storage or retrieval by the key of the data. This uses two techniques; one uses hashing and the other an index. Besides these techniques, the inverted index may be used so that certain data can be retrieved by the content of the data. Second is the sequential access method. It uses the order in the SET between the RECORDs, or the key sequence order in the index or the inverted index, or the stored order in the RANGE.
The internal access method used in AIM is a high-speed DAM which is an enhanced version of BDAM. This reduces both the path length of READ/WRITE instructions and the number of I/O operations. The path length is shortened by the simplification of BDAM and a dedicated basic IOS(I/O supervisor) driver. The reduction in path length averages 50% compared to BDAM. The reduction in the number of I/Os is obtained by accessing multiple blocks in a single read or write request. For example,there may be several blocks(up to 9) to be written back to the DASD. At the end of a transaction, these blocks can be written back to the DASD by a single write instruction.

6. Data integrity

AIM supports the following standard functions to guarantee data integrity.
- Exclusive control: Once a transaction starts, accessed data is exclusively controlled until the transaction finished. When data access requests are made in the transaction, the ranges, subranges or pages of the database to which the data belongs are temporarily locked. Upon termination of the transaction, the locks are released. The detection and release of deadlocks and the restart of application programs are carried out automatically.
- Log collection: Various log buffers and log files are provided for prompt recovery from various abnormal conditions. Before image data and after image data are available as log data for recovery. The before image data is used for backout when an application program terminates abnormally or a deadlock occurs. The after image data is used for recovery processing of a dataset in the event of DASD or system failures. The before image data is stored in a backout file(BOF) buffer provided for each task and written out in a BOF when the BOF buffer becomes full during a transaction. The after image data is stored in a task log buffer provided for each task and moved to the history log(HLF) buffers common to the whole system, when the task log buffer becomes full or at the end of a transaction. Subsequently, the contents of the HLF buffers are written out in the HLF. At this time, a high-speed temporary log file(TLF) may be used to write out the data upon completion of a transaction to

reduce the overhead in writing to the HLF. The TLF is accessed at least once for each transaction, and the reduction in TLF access time is important. In the TLF, all blocks of the same track have the same key, so that the rotation latency to write the after image data to the TLF is minimized. That is, when the after image data is stored on the track of the TLF, any blocks of the track can be written.

- Recovery and Restart Processing: Recovery processing differs depending on the cause of the abnormal state. AIM uses log data to recover from crashes of the system, data set and tasks and from deadlocks. The following data are secured on the DASD for use in recovery processing.
- transaction process table(TPT), in which the transaction processing information for each task is maintained.
- system checkpoint, for managing all the TPTs of the system.
- resident database checkpoint, in which the load/unload information for the database is maintained.
- privacy control: Each AIM resource(e.g. a work station, message queue node, sub-SCHEMA) is registered with the RACF(Resource Access Control Facility) which contains the security information. Security checks are performed at message distribution to the application program and at job initiation. Further, certain record types and/or set types can be specified in ADL with various constraints(e.g. no update) so that sub-SCHEMA resources can be protected from unexpected access from application programs.

Conclusion

Recently, the increase in the variety of requirements for DB/DC systems has become considerable. Since late 1976, AIM has been improved to meet new requirements. The system has been installed at approximately 1860 sites.

References

1) "CODASYL Data Description Language Journal of Development, June 1973", NBS Handbook, 113, 1974.
2) "CODASYL COBOL Committee Journal of Development 1978", The Secretariat of the Canadian Goverment EDP Standards Committee, 1978
3) D.D. Chamberlin et al.: "SEQUEL2: A unified Approach to Data Definition, manipulation, and Control", IBM J.Res.Develop pp. 560 – 575, 1976

Extended Data Management System (EDMS)

Kazuo Taguchi

Systems Engineering Department
Mitsubishi Electric Corporation
Tokyo, Japan
03-434-9838

1. introduction

EDMS (Extended Data Management System) is a general-purpose database management system of Mitsubishi Electric Corporation which was first released in 1977. It supports the network model of data and its data definition language (DDL) and data manipulation language (DML) are based on the CODASYL DBTG proposals. EDMS offers several special features, which include location mode INDEXED as the 4-th location mode, multiple inverted indexes, data-item level password, checksum computation for every page in the database, and data enciphering. However, this paper will focus on some recent activities in extending EDMS. They include automatic generation of DDL and DML, and a database engine for data manipulation.

2. automatic generation of DDL and DML

EDMS users have found that the CODASYL-based DDL and DML are difficult to learn and that they need an inordinate level of experiences for a good design and manipulation of the database. For example, they have found it very cumbersome to have to select the location mode of a group, define the set-selection mode and the set order attribute for each set, in addition to defining the database item, group and set. It became clear to us that casual users of EDMS needed some database design aids. We believed that automatic generation of DDL and DML statements will substantially enhance the flexiblity of EDMS for not only the casual users but also the database administrator and application programmers.

This has led to our development of MADAM (Mitsubishi Advanced Design Aid for database management) as an extension to EDMS. MADAM provides two major functions that facilitate database restructuring and reorganization. One function supports a menu-driven operator interface to generate customized DDL for data definition. Fig. 1 shows an example diagram similar to the Bachman diagram for expressing a database schema. MADAM allows the user to define such a database-schema diagram interactively on a CRT display and translates it into a sequence of DDL statements.

Another function is the automatic generation of DML statements. The user of MADAM may fill in system-generated menus to interactively specify the functions of a data-entry program. Fig. 2 shows an example menu that may be used to define a data-entry program for TEAMG in the schema diagram of Fig. 1. MADAM generates a sequence of DML statements corresponding to the menu the user fills in by inserting appropriate DML statements into skeletal COBOL programs stored in the system.

EDMS provides a casual-user interface, called IDP (Interactive Database Processor). IDP and MADAM are integrated into a total interactive database processing system, called TIDPS, to support the database users during the entire life cycle of the database.

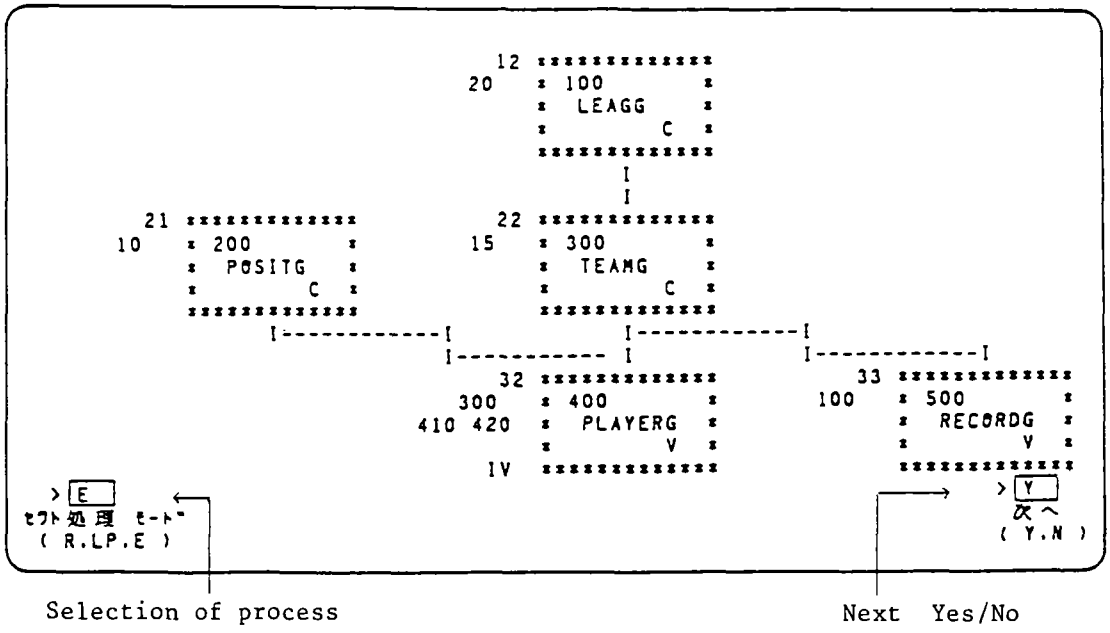


Fig.1 An example of a diagram for DDL generation

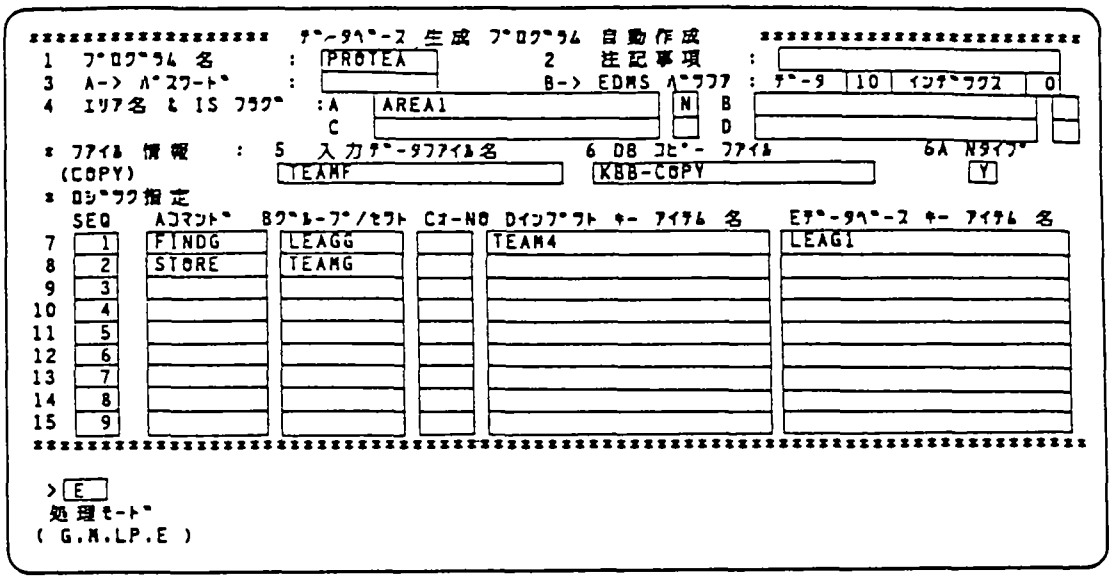


Fig.2 An example of a table to specify a data entry program

1. Program name 2. Comments
3. A - Password
   B - EDMS Buffer Size Data / Index
4. Erea name and IS flag
5. Name of input data file
6. Name of DB copy file

7 - 15 Definitions

3. database engine

Mitsubishi has incorporated functional engines into its MELCOM-COSMO 700/800 series computer systems. A functional engine is a high-speed wired logic which performs frequently used program logic. As one of the functional engines connected to the CPU, the database engine executes some EDMS functions. These functions include testing for data errors in a data page, retrieval of a record in a page, and transfer of data from the EDMS input buffer pool to a work area in the application program. Fig. 3 shows the performance improvement resulting from the use the database engine. Fig. 3.a shows the CPU time for the execution of the EDMS functions with and without the engine. Fig. 3.b compares the performance of three application programs. Application program A retrieves 10,000 records sequentially, program B deletes 1000 records after retrieving them using hashing, and program C deletes 2000 records after retrieving them using a primary-key index.

acknowledgements

I wish to thank Dr. Won Kim (IBM Research, San Jose), Prof. Y. Kambayashi (Kyoto University, Kyoto, Japan) and Dr. Mizoguchi (Mitsubishi) for encouraging me to write this paper and offering me many helpful comments on an earlier version of the paper.

Fig.3(a) CPU time comparison in execution of EDMS function with/without engine

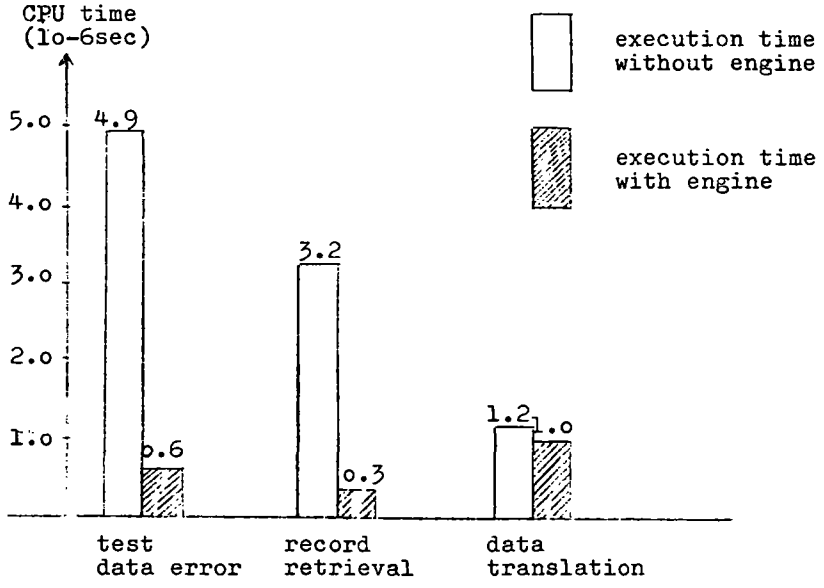
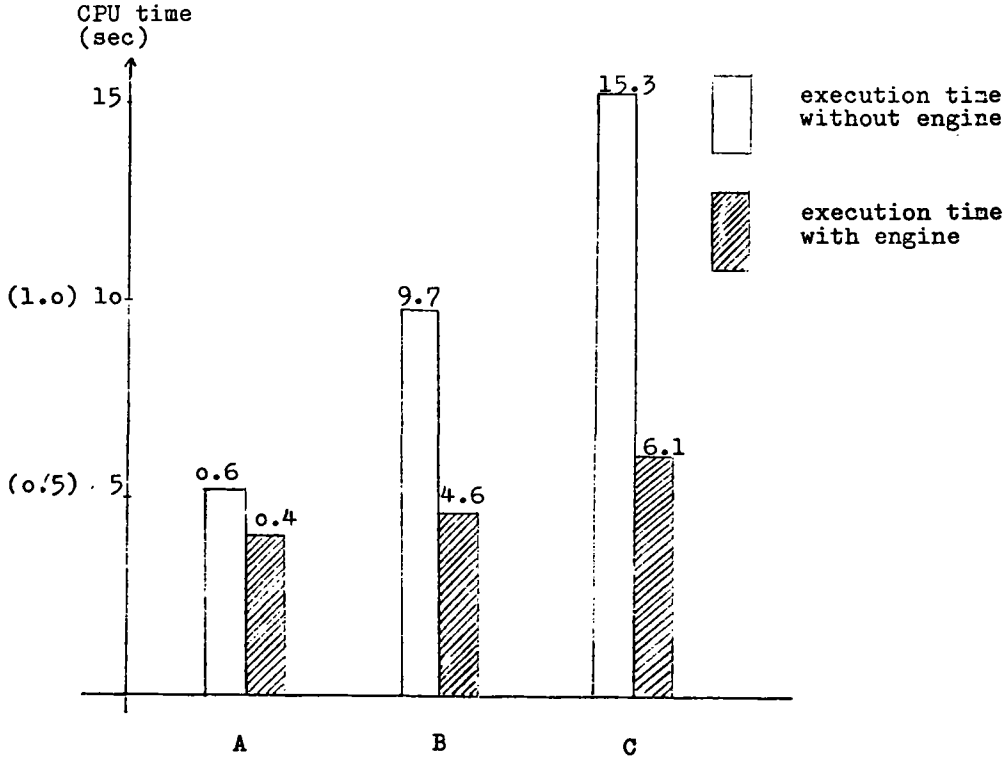


Fig.3(b) performance comparison for three application programs



ONLINE DATABASE MANAGEMENT SYSTEM

Kazuaki Masamoto
Software Works, HITACHI, Ltd.
5030 Totsukacho, Totsukaku, Yokohama 244, Japan

1. Introduction

DCCMII (Data Communication and Control Manager II) was developed to provide advanced data base/data communications capabilities for large scale computer systems. The system can be used in fields which require nonstop, quick response and high traffic online operations. Production control system in the steel industry is one of the typical application fields. The system is also available for general purpose online processing in various kinds of applications. The objectives of DCCMII is as follows:

1) To provide a DB/DC system with nonstop, high-performance, operator-free operation, and quick recovery-restart capability
2) To relieve users of the annoying complexity of on-line processing in areas such as multi-processing, re-entrant coding, handling of remote devices, recovery-restart, etc.

DCCMII supports database back-up and restore processing during online operation. And it also allows system modification without stopping online processing. The **MPP** (Message Processing Program) is executed in a single task, multi-partitioned environment so that it can be coded as easily as batch programs by using high level languages.

2. System Configuration

The system runs in a multi-partition environment under multi-virtual operating systems. One partition, called the control partition, is reserved for DCCMII control program, and the remaining partitions are used by MPPs (See Figure 2.1).

An MPP is loaded into an MPP partition and executed when a corresponding transaction is entered from a remote terminal. MPPs are expected to process the transactions in single task operations in multi-partitions. They issue subroutine calls to get messages from terminals, to access database, and to send replies to terminals. An MPP partition is serially reused by MPPs. Re-entrant coding and multi-task processing are not required for the MPPs. In the control partition, DCCMII allocates a subtask corresponding to each MPP partition in order to process those subroutine calls concurrently. Since an MPP is executed in an independent partition from control program and the other MPPs, the effect of an MPP failure is localized within the partition, and the other programs can continue their processing with no interference resulting from that failure.

DCCMII is configured with functional components which are divided into three groups, LEVEL I, LEVEL II and LEVEL III. LEVEL I components support all hardware dependent processing and the operating system interfaces.

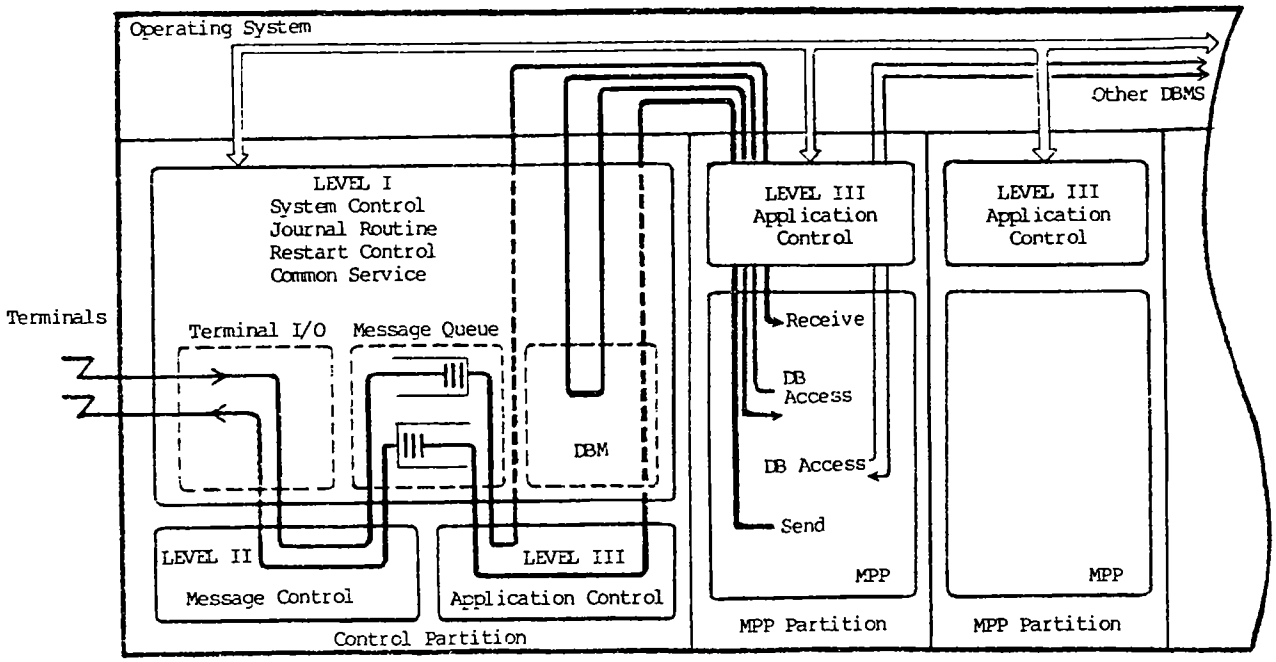


Figure 2.1 System Configuration

DCCMII task control and common service routines are also included in LEVEL I. LEVEL II and LEVEL III are transaction processing subsystems under LEVEL I. LEVEL II components control input and output message flows. And LEVEL III components support all application program related processing. An independent database management program(DBM) resides in LEVEL I, and it is called by MPPs via LEVEL III components. LEVEL I may be used by other subsystems. File transfer programs and other terminal-handling user programs are included in the subsystem.

3. DCCMII Message Processing

3.1 Message Routing

Messages received from terminals or application programs are inserted into the corresponding queues for processing. The output messages to be sent to terminals are queued on the corresponding logical terminals. A message destined for an application program is called a **transaction**, and it begins with a transaction code. The transaction code indicates the application program to be scheduled to process the transaction. Transactions are classified into several groups, and all the transactions in the same transaction group are queued in a serial chain based on the time of receipt by DCCMII.

Input messages are stored in virtual memory. Output messages may be stored in virtual memory, a message queue dataset, or in both. Output messages are classified into three groups; inquiry response messages, high priority switching messages, and normal priority switching messages. A message sequence number is given to each output message so that it can be identified when it is required to be re-transmitted. This can be applied to message recovery processing between two computer systems at system restart. Figure 3.1 shows the message queues.

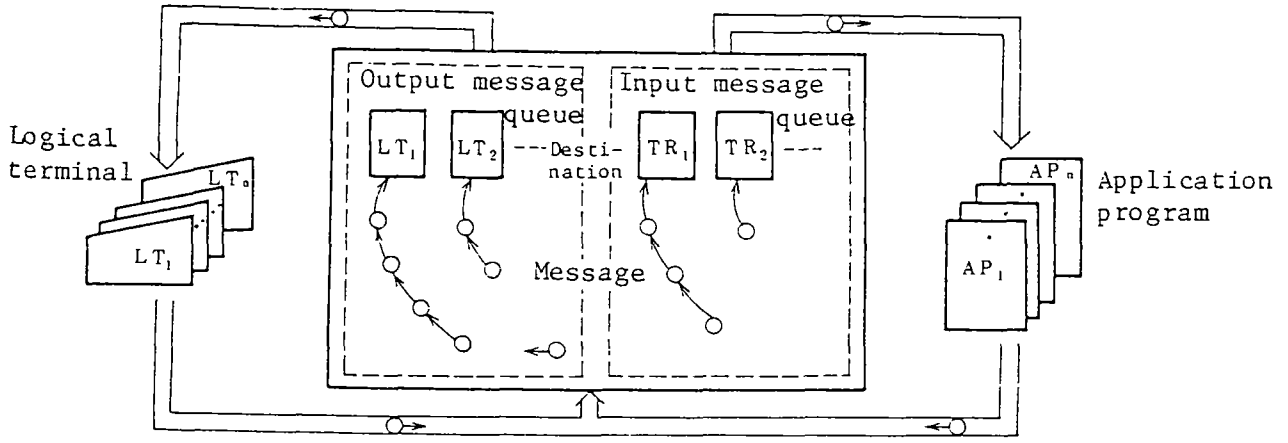


Fig. 3.1 Message Queues

3.2 Message Scheduling

When an MPP partition is started, up to four classes are assigned to the partition according to the initiation parameters. The partition classes declare that the partition is available to the transaction groups which belong to one of the classes assigned to the partition. The class assignment to a transaction group is done at system generation time.

The classes are used to improve the processing capacity of the system by balancing the amount of traffic, required memory size of MPPs, and so on. Exclusive control is performed by DCCMII. However, if the same class is assigned to transactions which update the same database record, those transactions are put into the serial queue for that class, and serially scheduled. Therefore, the database record is serially accessed, and the overhead associated with exclusive control of the database manager can be avoided.

When a transaction is chained in the input message queue waiting to be processed, and an MPP partition is empty, the application program designated to process the transaction will be loaded into the partition for execution. Of course, the transaction class and the partition class must be identical. If they are not identical, the partition is kept empty until transactions of identical classes arrive. The transactions in the input queue are kept waiting until an MPP partition with an identical class becomes empty.

4. Data Base

DBM supports two types of databases, MSAM and CHDAM. MSAM, Main Storage Access Method database, is a direct access database stored in main storage. At system start up, the database is loaded from an external device, and it is saved in an external device at system termination. It gives fast response and can be used for common area or control tables between application programs. CHDAM, Customized Hierarchical Direct Access Method database, is a hierarchical tree structure database using a VSAM ESDS dataset. The database structure and its access methods are simple and efficient, so DCCMII can be used in high traffic, fast response, online database applications.

DBM is called by MPPs via DCCMII LEVEL III in the same manner as message requests. It shares the system journal dataset, checkpoint dataset, and other system resources with DCCMII. Database update record information is stored in the DCCMII system journal, and is given back to DBM when an MPP terminates with a failure. DBM does backout processing on the database which was updated by the MPP, according to the database update record information.

PDMII* and ADM** are also available. They run in separate partitions from the DCCMII control programs. MPPs issue subroutine calls to access those databases. Requests are given to those database control programs directly from the MPPs. Synchronization and transaction recovery are performed by the DCCMII dependent partition control program which resides in the MPP partitions. System recovery is performed by the DCCMII control program, which requests the database control programs to backout or recover the database. This function is available to other stand-alone database control programs.

5. System Operation

5.1 Automatic Command Scheduling

DCCMII features the ACS (Automatic Command Scheduling) function. The system can be operated with fewer operator instructions, or even in operator-free operations. The standard operations can be stored in the ACS library, which includes system start-up procedures, system stop procedures, and procedures needed at specified intervals of system operation or at a specified time. These requests are analyzed and processed by DCCMII when specified conditions are satisfied.

5.2 System Re-configuration

DCCMII supports dynamic allocation of system resources so that the system can be re-configured during online operation. New transactions, application programs and terminals, for instance, can be introduced while the system is running. Even if the system terminates due to an unexpected hardware error or power failure, DCCMII simulates the modifications during restart processing by using the system journal, and the last configuration of the system is guaranteed.

The testing of new application programs at a remote terminal during online processing is available. The terminal operator declares that the terminal is in testing mode, or specifies the programs to be tested, and enters the corresponding transaction from the terminal. According to the specifications, DCCMII processes the testing of specified transactions and restores the updated system resources when the test is ended. The processing trace and the input/output data of the application program can be saved during testing.

---

* **PDMII**, Practical Data Manager II, is one of the most popular DBMS provided by Hitachi, Ltd. It supports network structure oriented database and limited hierarchical data structures.

** **ADM**, Adaptable Data Manager, is a general purpose large scale Data Base / Data Communication sytem provided by Hitachi, Ltd. The database of ADM is based on hierarchical data structures with logical relationships between database segments.

DCCMII/NOF (DCCMII Non-binding Online Restart Feature) supports the conversion of the current configuration and status of system resources to the new version of DCCMII. DCCMII/NOF saves all the information of the system into the DCCMII journal dataset at system termination. Since the control blocks and the record contents of new system is different from those of the old system, DCCMII/NOF interprets the saved information and gives it in a new form to the new system at restart processing. DCCMII/NOF allows for rapid installation of new versions of large scale online database systems in few minutes.

5.3 Database Recovery during Online Operation

DCCMII supports online database backup and recovery functions. The backup utility saves the current database during online processing, and the database change accumulation utility accumulates all the database update records from the system journal dataset. When the database crashes and there is no way to backout, the database recovery utility is used to restore the database using the backup database and the database update journal records. It simulates all processes of the online operation on the database, giving the same status of the database at the end of the system journal. This is performed in a batch environment. The database is returned to online processing without stopping the online operation. During database recovery processing, the online processing of the transactions which access the database are postponed until the database is recovered. The transactions which do not access the database are executed without any interference(See Figure 5.1).

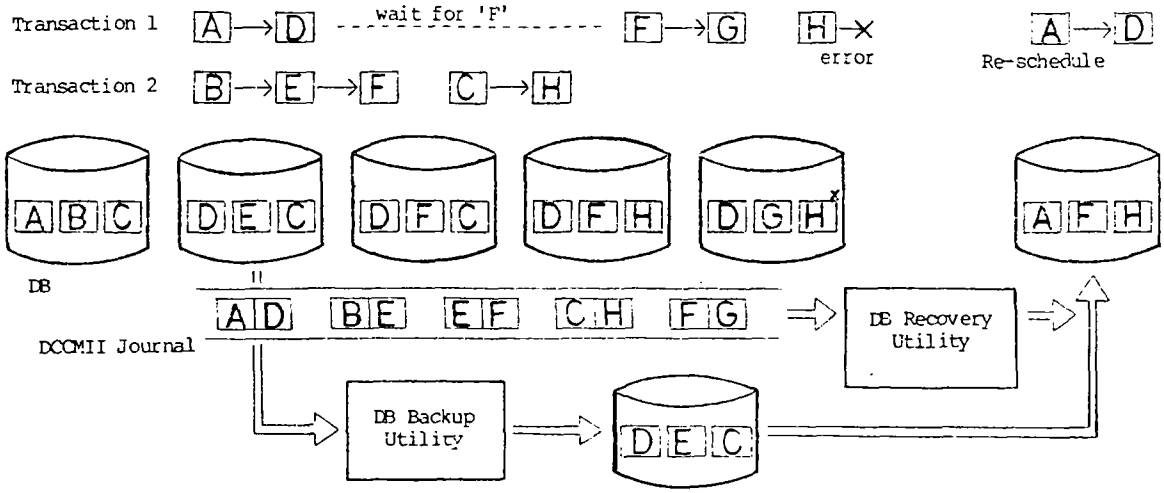


Fig. 5.1 Database Recovery

6. Current Status

The first version of DCCMII was released at the end of 1981. The practical applications of nonstop online operation using DCCMII are scheduled for the middle of 1983.

7. Acknowledgments

The author wishes to thank Dr. W. Kim, Dr. Y. Kambayashi, Mr. G. Rountree and Mr. Y. Ueda for their very useful comments and suggestions that improved the presentation of the paper.

Relational Data Base Systems

INQ and RIQS

Kazuyuki Hara, Tatsuo Gotoh, Tsuyoshi Miyazaki, Ken Takeuchi and Shigeru Mabuchi

Nippon Electric Co., Ltd.
NEC Building, 33-1, Shiba Gochome, Minato-ku, Tokyo 108, Japan
Tel: Tokyo (03) 454-111

Introduction

NEC has developed two relational data base systems; INQ (INformation Query) and RIQS (Relational Information Query System). INQ is a relational type data base system with characteristic data structures, and RIQS is a relational data base system of high degree of sophistication which has been developed as a sequel to INQ. Ease of use for end users was emphasized most in RIQS. The end user language (EUL) allows the user not only to obtain answers but also to edit them to prepare reports and graphs. This paper describes the design concept common to both systems first, and then the characteristics of each system.

It should be noted that such functions as concurrency control, data integrity, data security, failure management, etc. of INQ and RIQS have been integrated into the operating system, and concurrent multiple updates, high responsiveness, complete recovery and durability are guaranteed even for large data bases.

1. Design Concept

INQ and RIQS emphasize ease, flexibility and responsiveness in designing and operating data base systems.

1.1 Ease of Design and Operation

In building a data base system, it is necessary to design and define a data base structure and store data. A high level of knowledge of data base is required to conduct such activities using conventional data base systems. In INQ and RIQS, however, the structure of the data base is simply based on a table (which defines a relation), and it is possible for programmers and end users to directly design and define data structures of their own. Each table can be defined with a knowledge not more than that required for preparing a sequential file, and it is not necessary to consider the storage structure which is usually very complicated. Tables can be defined in batch mode or interactively under TSS, and it is possible to define them dynamically (see Figure 1).

Once a table is defined, the system automatically provides the user with the access right as the owner of the table. It is also simple to assign other users with access rights to the table. It can be conducted interactively using a guide for defining the access right as in the case of defining a table.

Fig. 1 Sample Table Definition in TSS (RIQS-EUL)

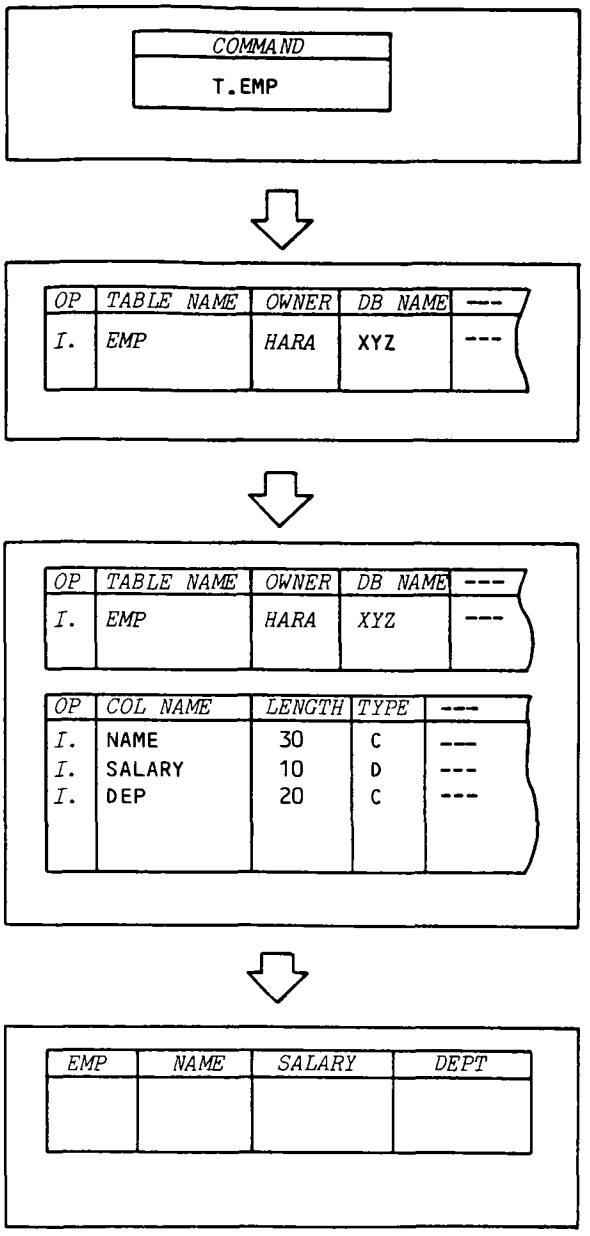


Table definition command and table name are entered from keyboard.

A guide table for table definition is displayed, and table attributes are entered.

A guide table for column definition is displayed, and column names and their definitions are entered.

The final table format is displayed when all definitions are complete. The table name (EMP) column is used to enter data proessing operators such as P (Print), U (Update), I (Insert) and D (Delete).

Note: Information entered by end user from keyboard is indicated by OCR font.

Data can be stored by using the insertion function provided as one of the query operators. It is also possible to store data at high speed using a utility function of the system.

Programming languages applicable to relational data bases are high level languages such as COBOL, PL/I and FORTRAN and respective end user languages. A set of records is manipulated by one instruction instead of one record occurrence. This feature not only simplifies the logic of application programs but also allows the use of relational operations which are not applicable on conventional data bases.

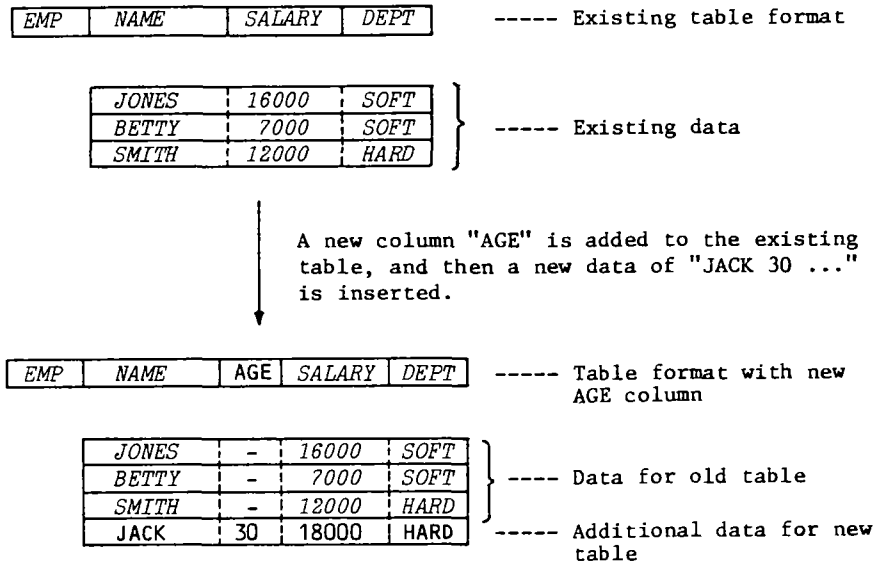
End users often find it necessary to edit answers or express them in graphic representation. It usually takes too much time to modify answers into input data for some dedicated programs. RIQS is provided, however, with functions for preparing business graphs along with powerful editing functions, and end users can develop and operate application systems using the RIQS data base system.

1.2 Flexibility

The following functions are provided in order to minimize the impacts of changes to the database.

(1) New columns can be added to an existing table quickly from a terminal without disturbing existing data, and both new and old data become available for processing. It is not required in such cases to modify existing programs and queries (see Figure 2).

Fig. 2 Addition of a New Column to an Existing Table (INQ and RIQS)



(2) It is possible to combine existing tables through a common column (JOIN function) into a virtual table. The virtual table thus obtained can be treated just like a new table except that such a table is used only for retrieving existing data. This function allows the user to process data on different tables without creating a new table, as if they were on one table.

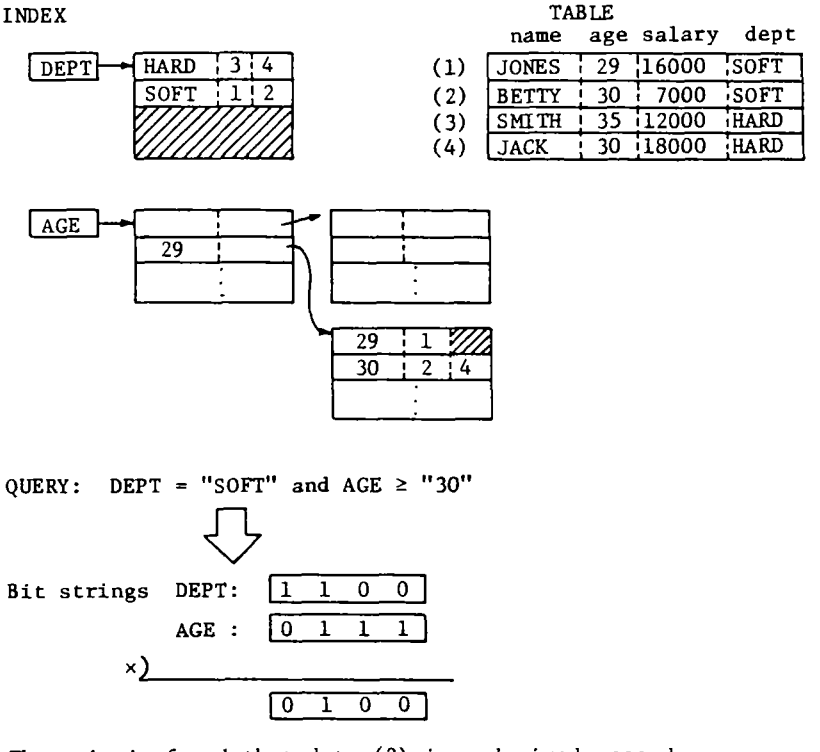
(3) Any columns, indexed or otherwise, can be used as search keys in retrieving data. This feature allows to create a search pattern independently of the storage structure of the data base.

(4) RIQS allows to define ADBS subschemas by tables of form of the virtual table of the relational data base, and the user can retrieve data of network structure directly through such tables.

1.3 Responsiveness

One of the main objectives of a data base system is to allow the user to retrieve necessary data speedily from a large volume of data. In the INQ and RIQS systems, indices are dynamically generated for columns specified as search keys. Such indices have a B-tree structure, and each index basically contains the values of records for the column and the corresponding record sequence numbers. When conditions are specified on two or more columns, for example, their respective indices are searched, and the record sequence numbers are converted for each index to a bit string consisting of 1s and 0s corresponding to TRUE or FALSE. A desired record is obtained by performing logical operations on such bit strings (see Figure 3).

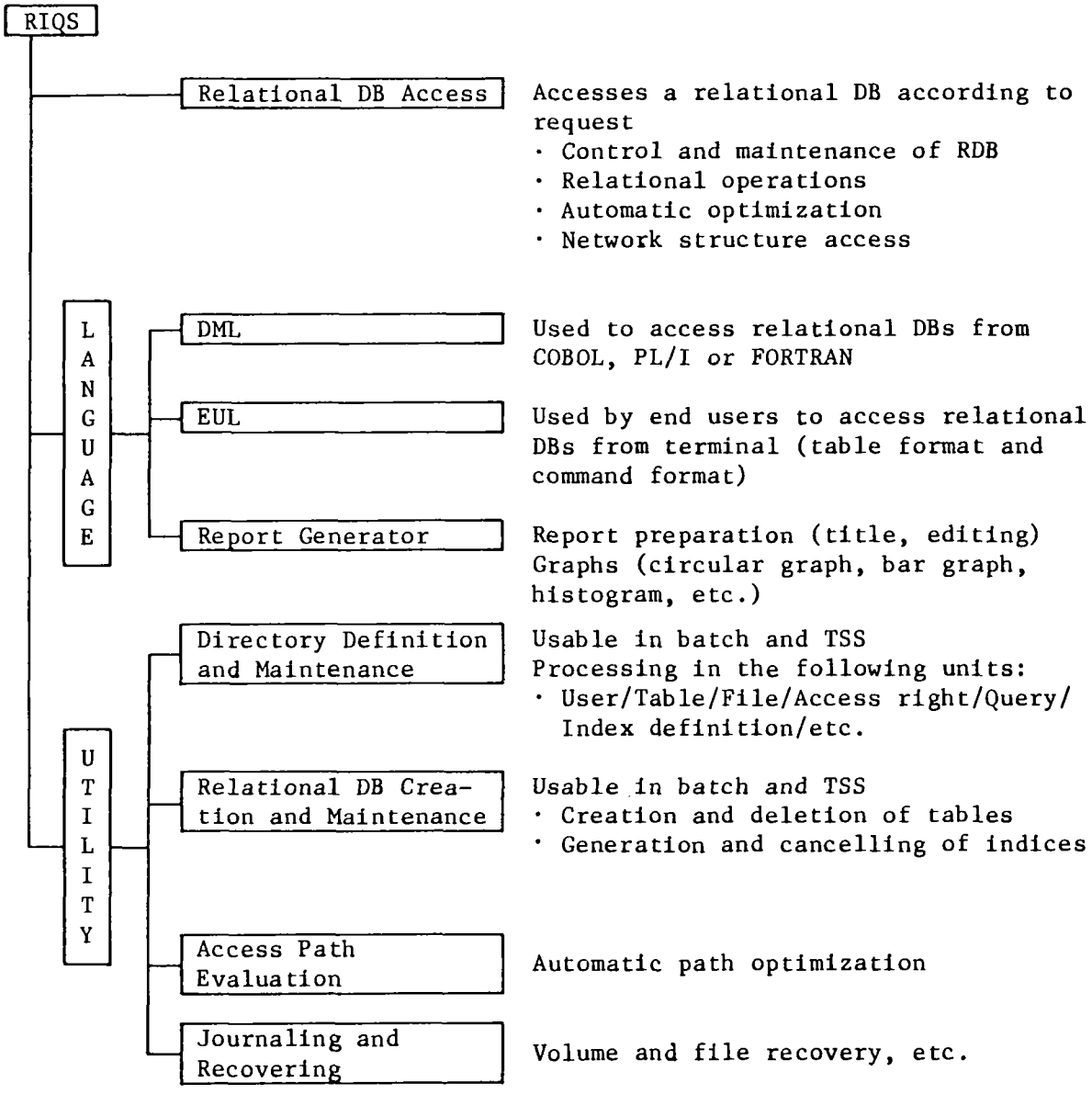
Fig. 3 Indices and Bit Strings (INQ, RIQS)



Thus, it is found that data (2) is a desired record.

The software organization of RIQS is shown in Figure 4.

Fig. 4 RIQS Software Organization



2. Data Structure of INQ

The relational data model has contributed greatly to the development of data base management systems. From a practical point of view, however, there is considerable bewilderment on the part of the user because many relations are created to satisfy data normalization and it becomes difficult to determine which relation to use. In order to resolve such problems at a practical level, the relational data model has been extended as described below for INQ.

The first normalization of data evolves to the second and third normalization in order to eliminate update anomalies. Problems caused by such evolutions include the following:

(1)Same data appear in two or more relations due to division of relations, and data must be updated in all such relations (see man# in Figure 5).

(2)When a relation is divided, programs which reference the relation must be modified, compromising data independency.

In INQ, it is allowed to assign multiple values to one field. That is, INQ admits unnormalized data structures to some extent. Thus, INQ is usable for processing business data, descriptive data and statistical data containing repetitive groups (see Figure 5; INQ is capable of handling both types of data).

Fig. 5 Example of Normalization

(a) Unnormalized data structure

```
EMP (man#,name,project#,projectleader,
   JOB (date,title,SALA(month,salary)),
   CHLD (childname,age))
```

(b) Normalized data structure (3rd normal form)

```
EMP (man#,name,project#)
JOB (man#,title,date)
SALA (man#,month,salary)
CHLD (man#,childname,age)
PROJECT (project#,projectleader)
```

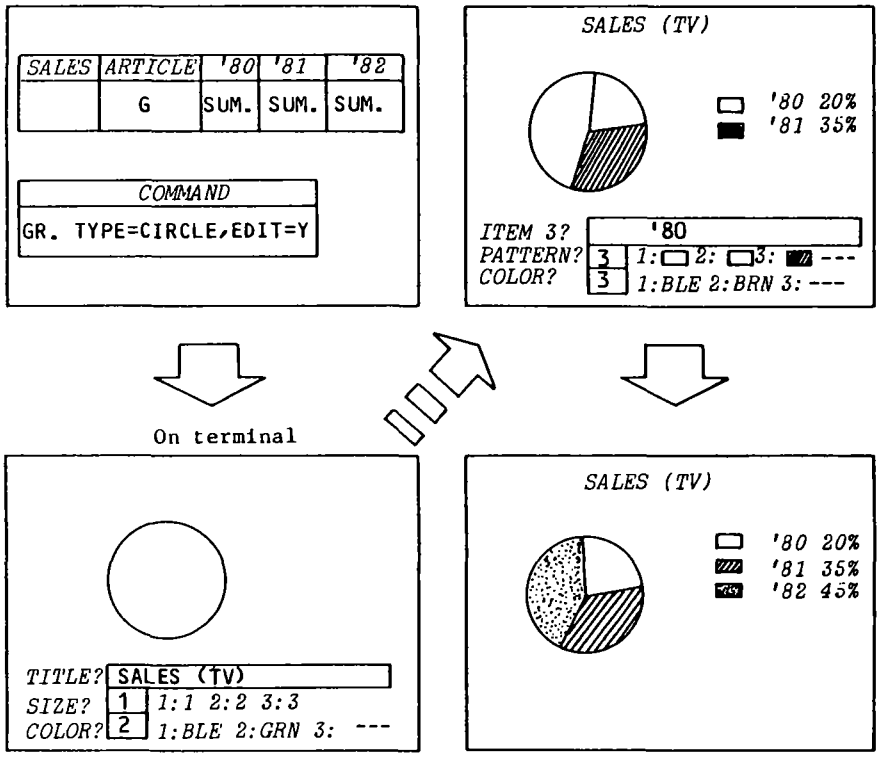
3. Functional Distribution in RIQS-EUL

Intelligence in terminal equipment has been improving rapidly, and it has become possible to equip most terminals with graph processing capabilities. In order to take full advantage of these functions of terminal equipment, RIQS-EUL performs graph processing not on the host computers but on the intelligent terminals. This arrangement offers the following additional advantages:

- Reduction of load on the host computer due to distributed processing
- Reduction of data transmissions between the host computer and the terminal

An example of circular graphing (RIQS-EUL) is presented in Figure 6.

Fig. 6 Example of Circular Graphing (RIQS-EUL)



4. Concluding Remarks

The characteristics of relational data base systems are that the data structure is simple and the data is manipulated in the unit of data set. Thus, it is possible for end users to understand data access procedures and describe them by themselves.

For INQ and RIQS, end user languages which allow end users to program by themselves have been developed. The use of such languages allows to reduce the overall amount of programming, greatly enhansing the program productivity.

It is planned to strengthen the following functions in order to make these relational data base systems much easier to use:

- o Improvement of end user languages to allow to respond to various requirements of end users in a variety of fields (development of functional routines for various fields)
- o Distribution of host computer functions with the effective use of capabilities of intelligent terminals (syntactic check and graphics)
- o Automatic optimization of access path to guarantee a consistent level of performance against the increase in the amount of data.

Note that the INQ and RIQS relational data base systems operate under the control of the ACOS-2/-4/-6 operating system.

INTRODUCING ADABAS TO THE JAPANESE MARKET

Yoshioki Ishii
Software AG of Far East, Inc.
2-7-2 Yaesu, Chuo-ku, Tokyo

1. Introduction

ADABAS (Adaptable DAta BAse System) was initially designed and programmed in 1969 by P. Schnell* of Software AG, West Germany and Version 1 was completed in 1970. The performance and facilities of ADABAS have been enhanced extensively over the years. In 1974 ADABAS was first installed in Japan, and now the number of installations in Japan and the Far East exceeds 100.

No DBMS on the market other than ADABAS and some of the mainframers' DBMS's (such as IDMS, System 2000 or M-204) has been successful in Japan. We believe that it is mostly due to the fact that the vendors cannot provide satisfactory technical assistance and consultation for the users, compared to Software AG and the mainframers.

The present paper first describes the background of ADABAS development. Then the software architecture of ADABAS is explained. Finally, our efforts to develop additional facilities for ADABAS are described.

2. Background of Development

We consider it very important and interesting that the relational model and ADABAS were developed quite independently, but ultimately shared the same basis.

The common basis is the non-procedural data search and data and program independence. However, ADABAS fully utilizes the inverted index. The use of inverted index technique is also very useful to enhance JOIN of the relational model.

Objectives of ADABAS

ADABAS was designed to address the following potential problem areas:

- To enable the use of programs without modification, even if data type and length are modified, the field ordering in records is modified, and new fields are added to records thereby causing the record length and layout to change.
- To reduce data storage space by removing null value fields, repetitions without values, trailing spaces of alphanumeric items, leading zeros of numeric items, etc.
- To enable the entry of data longer than a predefined field length.
- To enable the use of multiple direct access keys for a file.
- To enhance performance of content searches on data which are separately stored in many files (JOIN).
- To reduce the use of sort and merge operations.

It was 1970 when Version 1 was completed, meeting these basic requirements.

---

* P. Schnell is president of Software AG, Darmstadt, West Germany.

3. Architecture of ADABAS

The architecture of ADABAS data base, its user interface, and the DBA features are outlined in Fig. 1.

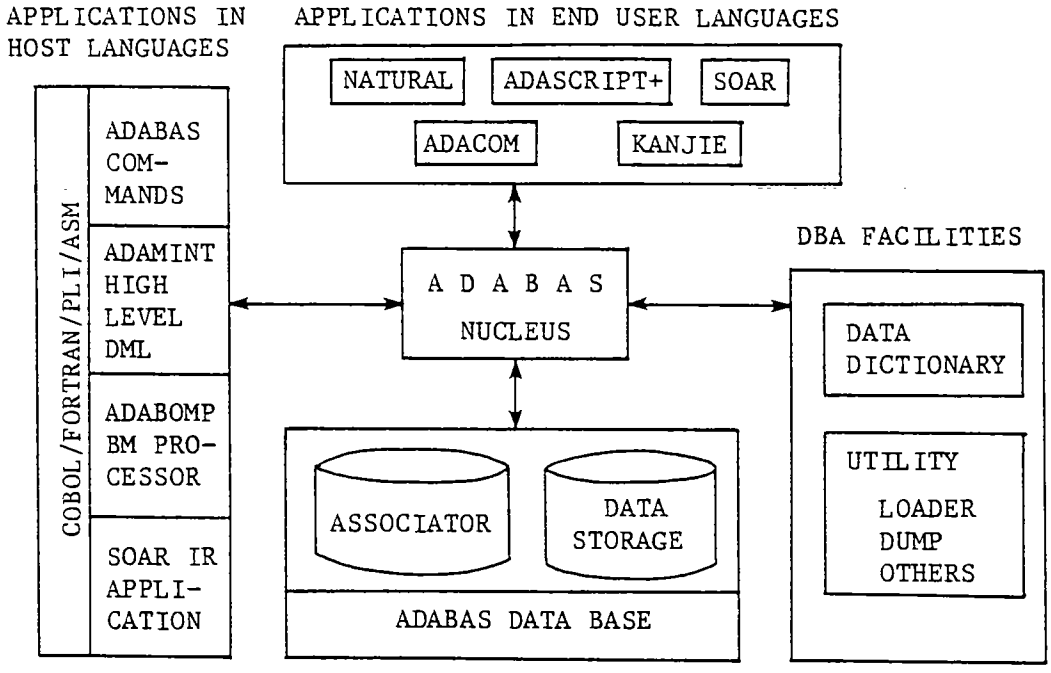


Fig. 1. Overall Architecture of ADABAS

The ADABAS nucleus processes user commands for data base access and update. The data is stored in a compressed form in DATA STORAGE. A collection of uniform records is called a 'file'. ADABAS can directly handle files (relations) which are not normalized. A relation of the relational model is a special case of an ADABAS file.

The ASSOCIATOR is an access facility which facilitates access to the data in the data base. It is composed of inverted indexes and coupling indexes (inter-record relationship indexes).

ADABAS has two user interfaces. One is for the application user and the other is for the DBA (Data Base Administrator).

The following two types of languages are provided for the application user:

a) End user languages (self-contained type) ---------- ADASCRIPT+ (Interactive language), NATURAL (Interactive programming language), SOAR (Interactive IR language), KANJIE (ADASCRIPT+ for a Kanji terminal).

b) Data manipulation languages (host language type) --- ADABAS commands, ADAMINT (Userview facility), ADABOMP (Bill of material processor), SOAR (IR commands).

The DBA interface comprises the following facilities:

a) File loader --- Utility program to load a file.
b) DBA support utilities --- Data base dump/restore, recovery, definition, modification, creation and deletion of inverted indexes and coupling indexes.
c) Data Dictionary --- ADABAS Data Dictionary.

4. ADABAS Enhancements in Japan

To meet customer demands in Japan, Software AG, Germany, added several new facilities to ADABAS. One of the most notable features is the "Parallel Dump". It allows concurrent data update and DB DUMP, and guarantees data integrity at the time of DB RESTORE.

Further, Software AG of Far East has developed ADABAS interfaces to Japanese operating systems and TP monitors, and the optional capabilities described below.

a) Online KANJI Report Writer KANJIE,
b) interactive IR Language SOAR, and
c) IMS/ADABAS BRIDGE.

The following sub-sections outline KANJIE, SOAR and IMS/ADABAS BRIDGE.

4.1 KANJIE

The ordinary terminal has been able to handle alpha-numerics only and for a long time Japanese users could not use regular Japanese characters (Kanji, Hirakana and Katakana) which we use. Japanese characters are expressed with 2 bytes/per character. Therefore, the interactive language based on such Japanese characters can truly be called a natural language for the Japanese. That is why we began to adapt the basic interactive language, ADASCRIPT+, developed for ADABAS to allow input and output in Japanese. This is KANJIE (KANJI Easy).

For the past few years, Japanese character (Kanji) terminals began to be installed at user sites. Roughly speaking, there are several thousands of Japanese characters which are in daily use. Therefore, the keyboard of the Kanji terminals is a huge one and the outlook of it is quite different, and the end user finds it difficult to use. In order to avoid such inconvenience, KANJIE adopted a method to convert 1 byte phonetical Kana character input into 2 byte Kanji, which we call Kana-Kanji conversion, and in this way the ordinary keyboard can be used for Kanji input.

4.2 SOAR

SOAR (Set Oriented Architecture of Request) can serve as a powerful user-friendly aid for developing information retrieval application systems.

SOAR supports several IR commands, which are activated by SOAR calls embedded in host languages. We have also developed an interactive SOAR language.

Two types of the techniques are available for retrieving necessary information. One is to utilize inverted indexes, and the other is to sequentially scan the data and pick up qualified records. Although ADABAS commands are used for the first type of retrieval, a new set of commands has been devised to support the second type of technique. SOAR users can also save retrieved sets of records. The saved set of ISNs (Internal Sequence Number = Tuple ID) can be manipulated by means of several set operations later on. ISN set created by executing SOAR commands can be saved in the ISN FILE. The ISN FILE is a system file stored in the ADABAS data base, and is maintained by SOAR.

In the design of the ISN FILE, we decided that, because of system overhead, the contents of the ISN FILE should not always be updated by updates to relevant

data. Although the ISN FILE is not always updated by SOAR, users can easily make its contents up to date. The key to this is the specification of construction rules for each ISN set in the ISN FILE. Then SOAR readily refreshes the ISN set based on the construction rules, if the user initiates a command for refreshing the ISN set.

4.3 IMS/ADABAS BRIDGE

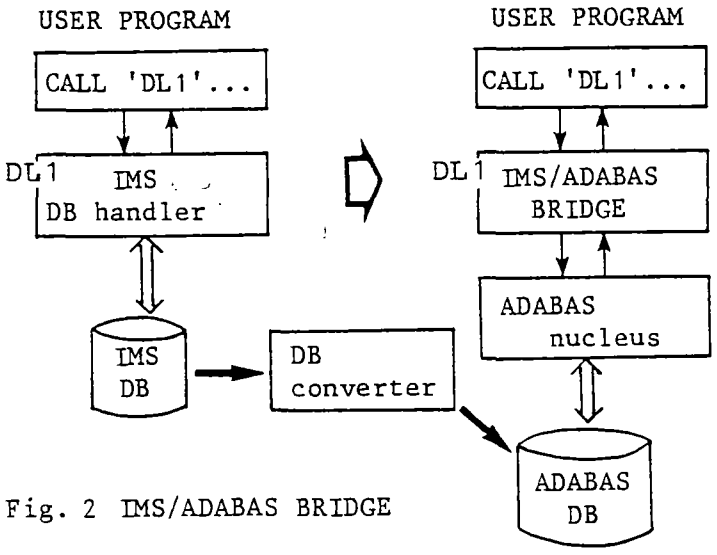


Fig. 2 IMS/ADABAS BRIDGE

IMS/ADABAS BRIDGE was developed in 1978, and now it is utilized not only in Japan but also overseas. It is an emulation module which enables the user to execute IMS application programs on ADABAS as illustrated in Fig. 2.

The DB converter reads the IMS DB in the hierarchical sequence and dumps it to the sequential file (see Fig. 3). Then it is loaded to the ADABAS DB by the ADABAS file loader. In this case, the data compression function of ADABAS performs an important role.

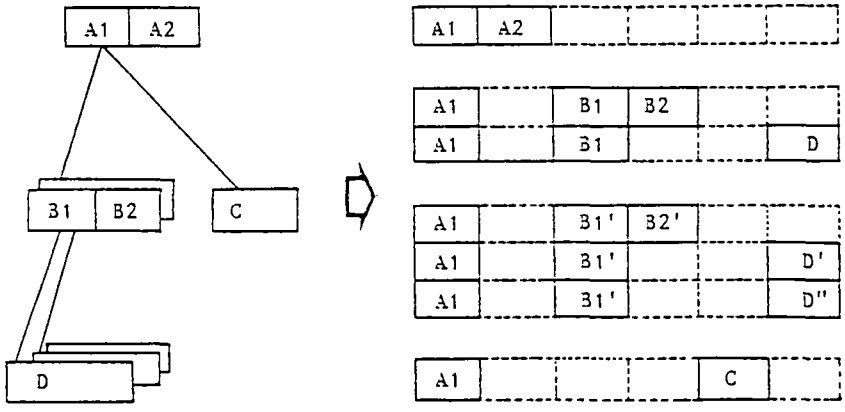


Fig. 3 Data Base Conversion

By adopting the BRIDGE, the user programs need not be changed at all. The BRIDGE accepts the DB CALL parameters of IMS as they are. So, if an error should occur, it is converted to the IMS error code and handed over to the user program. In this sense, this BRIDGE has a complete emulation function.

5. Concluding Remarks

The important factor to succeed in distributing software products in Japan is to fully understand the users' requirements and to continue enhancement to meet them. In addition, and a more important unavoidable factor is to provide the facility to fully handle the Japanese language. We endeavoured to fulfill these requirements and succeeded, but all the Japanese vendors who simply imported and tried to sell DBMS failed.

Overview of IMS/VS Fast Path Enhancement

Toru Takeshita
Software Development Center, IBM Japan
1-14, Nisshin-cho, Kawasaki-ku, Kawasaki, 210 Japan
Telephone: 044 - 201 - 2405

1. Introduction

IMS/VS (Information Management System/Virtual Storage) is a general-purpose DBMS, which is most widely used in the world. Its applications range from personnel information to the real-time production control. However, its functions have been so generalized that its performance is considered not to be satisfactory for some realtime applications even on the highest MIPS machines available in the commercial market. On the other hand, because of the increasing dependency of industrial, business and consumers' activities on the on-line computer system the data integrity and system reliability and availability have become highly critical factors of DBMS.

On-line banking systems in Japan are highly advanced in terms of the number of transactions per second and sophistication of data storage and processing. They will become even more so with the installation of the third generation banking systems in the middle of 1980's to provide more expanded customer services (including kanji and image data) with new offerings and management information for better planning and control of resources. It is predicted that the transmission volume to be handled by the DP center will reach as highly as 200 messages per second in 198x.

Banks are no longer interested in developing and maintaining their own (so-called Roll Your Own) huge on-line control program which may require hundreds of person years to develop and which must be continuously modified and expanded as new requirements, new devices and new software functions come up. To find a common solution to this for a large number of banks has been no easy task. A number of alternatives had been studied and proposed by the middle of 1980. The final choice was to change and expand the functions provided by the IMS/VS Fast Path, which had been originally developed for simpler on-line banking applications in North America and Europe, as well as to expand the IMS/VS base by adding DASD logging, 255 dependent regions (for concurrent execution of more application programs), on-line changes (of data base definitions, programs, transaction codes, display formats), etc..

The FP functions expanded by IBM Japan Software Development Center include the extended hierarchical structure, deactivation of a physical block (control interval) in error, updating multiple copies of the data base (more precisely data set area), and data sharing at area and block levels.

This paper describes those new functions (called Fast Path Enhancement) which have been developed in Japan (with support from IBM General Products Division's Santa Teresa Laboratory) as well as the original functions which were developed

in the first and second releases of the Fast Path feature of IBM IMS/VS. FPE was originally intended only for Japan, but was announced world-wide in October, 1982.

Functions described in Sections 3, 4, 5 and 6 are those which were developed in FP R1 & R2. And Sections 7, 8 and 9 contain new functions implemented in FPE.

2. Partial Overview of IMS/VS Architecture

This section is intended to give some explanation of the IMS/VS structure and terminology which are required to understand FPE.

IMS/VS is an IBM program product that runs under the OS/VS operating systems, and is offered as a basic system with optional features. The basic system, the Data Base (DB) system, provides facilities for defining, creating and maintaining IMS/VS data bases (hierarchical with logical relationship and secondary index facilities), and for running user-written programs in the batch-mode to process the data bases. The features include the following:

- Data Communication (DC) provides for the transmission of messages between IMS/VS and remote terminals, and for the invocation of user-written programs to process incoming messages againt IMS/VS data bases.

- Multiple System Coupling (MSC) provides for the routing of messages between two or more IMS/VS systems that run in the same or different CPUs.

- Fast Path (FP) provides message processing and data base facilities for high-volume, high-availability DB/DC applications. This was initially developed for IMS/VS 1.1.4 which was released in 1977, and was slightly expanded in 1978. The new/expanded functions implemented in the latest release (1.3) of IMS/VS are collectively called "Fast Path Enhancement (FPE)" in this paper.

- DASD logging provides logging of transaction messages and data base update information onto DASD devices. This has been developed as part of IMS/VS 1.3.

- Data Base Recovery Control (DBRC) manages resources for data bases, logging and back-up, etc., controls back-up/recovery, authorizes the use of data bases, and controls sharing of data bases. This is mandatory for multiple area data sets in DEDB and the shared use of DEDB.

In IMS/VS 1.3, MSC and FPE are integrated into the IMS/DC.

In this paper, "IMS/VS base" means IMS/VS DB and DC with DASD logging but without FPE, DBRC and MSC functions.

An on-line execution of IMS/VS uses one or more operating system regions. One of these regions, the control region holds the IMS/VS control program. The remaining regions are called dependent regions, and are used for the user's application programs. The latter are of three types: message processing regions and batch message processing regions both used in IMS/VS base, and message-driven regions used by FP - also called IMS/VS Fast Path Processing (IFP) regions.

IMS/VS data base functions are provided by a set of program modules called DL/I (Data Language/One). A data base can be accessed or updated by a DL/I call statement. This has the following format if used in an application written in PL/I:

CALL PLITDLI (no. of params, fn, DB, I/O area, seg. search arguments);

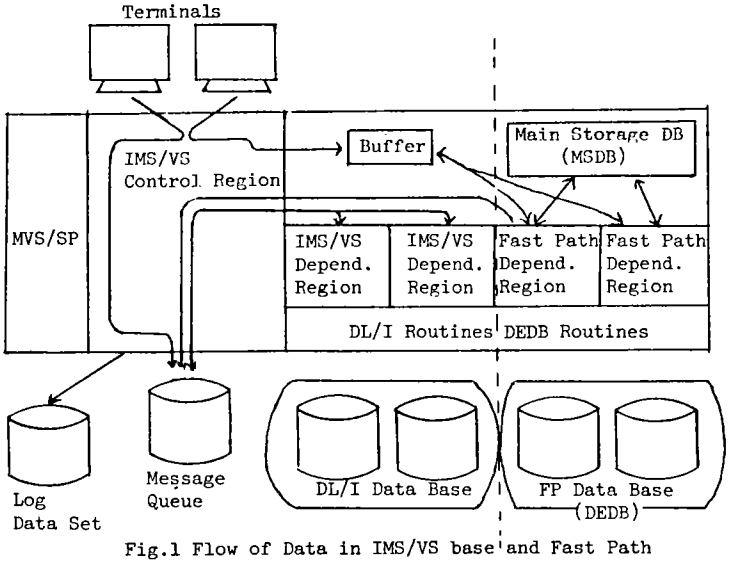


Fig.1 Flow of Data in IMS/VS base and Fast Path

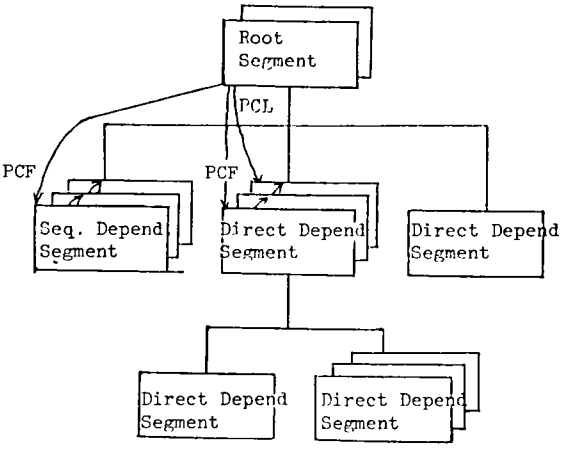


Fig.2 Structure of DEDB

In Figure 1, the IMS/VS base is without the portion to the right of the dotted line.

Fig. 2 indicates the data base structure discussed in Section 7. In IMS/VS, a segment is a minimum unit of data to be accessed by an application program. A logical record consists of multiple segments, each of which is made up of several fields. A direct dependent segment is accessed by means of a physical child first (PCF) pointer stored in the root (or parent) segment. Sequential dependent segments are stored sequentially and linked to the root segment by a LIFO chain.

A unit of data to be read from or written onto a data base by a physical I/O operation is a block, which is also called a control interval when VSAM (Virtual Storage Access Method) is used. When a physical error occurs in the I/O operation of a particular block in a DASD, FPE can make only this block inaccessible by the application program, which can continue accessing other blocks in the same DASD. To create this status is called block deactivation.

A Data Entry Data Base supported by Fast Path can consist of multiple VSAM data sets, each of which is treated by the operating system as a separated physical file handled by VSAM. These data sets in the DEDB are called area data sets or simply areas.

IMS/VS DC provides three message processing modes: response mode, non-response mode and conversational mode. In the response mode, the program must transmit a response message to the originating terminal before another transaction is accepted from that terminal. Fast Path has only this mode.

Fast Path application programs operate in the wait for input mode, which means that they are pre-loaded in the main memory and initialized ready to process messages upon receipt. They are interlocked when they attempt to retrieve from

any empty queue.

Multiple execution of a FP application program may be scheduled concurrently into different regions. All executions are associated with a single load balancing group, and retrieve transactions in FIFO sequence from the single transaction queue that is associated with the group. The number of program executions is determined by the system operator.

3. Expedited Message Handling

Expedited Message Handling enables fast message processing by allowing 1) each message to have only one segment or unit and a buffer to be associated with it for I/O messages, 2) the terminals to operate in the transaction response mode 3) messages to be processed in wait-for-input mode, 4) multiple copies of the same application programs to run concurrently, and 5) messages to be queued in a FIFO order in a balancing group prior to transfer to a particular application program.

An input message is stored in a buffer allocated to a particular terminal. Then, it is queued in the balancing group associated with the application program, which is to process it, The output message from the user program is stored in the terminal buffer to be transmitted to the terminal.

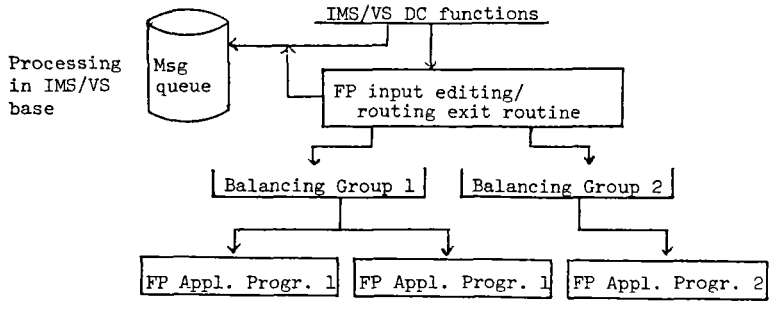


Fig.3 EMH Scheduling (example)

4. Data Entry Data Base

The Fast Path feature has its own data base called DEDB (Data Entry Data Base), which was originally designed for data entry-type applications. Its structure had only two hierarchical levels and contained up to 8 segment types. One of the dependent segment types can be sequential. Sequential dependent segments are used for fast collection of detailed information and are useful in journaling and auditing applications.

A DEDB can be partitioned into a maximum of 240 areas, (reaching as much as 960 GB as compared to 4 GB in an IMS/VS HDAM data base,) each of which contains a different collection of data base records. The data in a DEDB area is stored in a VSAM (Virtual Storage Access Method) data set. Each area data set is divided into the root-addressable portion, the independent overflow portion and the sequential dependent portion.

The unit of DEDB resource allocation within an area is called 'control interval'. A control interval can be accessed by only one application program at a time, while the control intervals in the rest of the area are available to other application programs.

5. Main Storage Data Base

Most frequently accessed data can be stored in the Main Storage Data Bases, which are 'root-only, fixed-length record data bases' loaded in virtual storage at IMS start-up. An MSDB can be accessed by a DL/I call and also by a special call, "field call", which allows access to a specific field within a segment and to change the value in that field independently of its current value.

MSDBs can be used for storing terminal control and statistical data (in case of the terminal related data base), for storing various kinds of tables (in case of the non-terminal related DB), for conversational processing work areas, and for storing user checkpoint information to re-start a batch-type program.

6. Commit Processing

When several FP application programs access a data base concurrently, FP makes it possible for them to read and update the data base and protect data integrity. This is done by preventing a program from accessing data that another program is updating until the updating program reaches a commit point. A commit point indicates that the program's data base updates thus far are valid; and that, if the program were to terminate abnormally at some later point, the updates it has made to the data base thus far are valid, and should not be discarded.

If the program terminates before the next commit point, the updates the program has made to the data base is discarded. A commit point is indicated by a call to retrieve a new input message.

Commit processing is done in two phases. In the first phase, the MSDB to be updated is enqueued (locked) and the availability of the DEDB to be updated is checked. In the second, input and output messages and data base update information are written out onto log data sets. After this, the updated information is reflected into the MSDB, and then requests to update the DEDB are made to the IMS/VS Control Region (, where these are done asynchronously). The enqueued resource (other than the VSAM control intervals to be updated) are dequeued and transmission of output messages is requested. This completes the processing of transactions in the dependent region, which can now start processing of the next transaction.

7. DEDB Extentions

In FPE, the DEDB structure has been expanded to 15 levels and 127 segment types, one of which can be sequential dependent.

Other extensions to DEDB include the physical child last pointer, which is performance option for the direct dependent segment type and provides direct access to the last occurrence of child segment types, thus giving increased similarity to HDAM (hierarchical direct access method) used in IMS/VS base. Also DEDB can now have subset pointers to point to subsets of chained dependent segments.

8. Multiple Copies of an Area Data Set and Error Handling

The data in a DEDB is stored in a VSAM data set. The user can create as many as seven copies of each area data set. They can reside on different devices, and even on different device types. When an application program updates data in an area, FPE updates that data in each of the area data sets. When an appli-

cation program reads data from an area, FPE retrieves the requested data from any one of the available copies of the area data set. This capability is unique to FPE, and does not exist in IMS/VS base.

If an error occurs while a DEDB is being updated, there is no need to stop the data base or even the area. FPE continues to allow the application program to access that area, and only prevents it from accessing the control interval in error. If there are multiple copies of the area, chances are that one copy of the data will always be available. If many errors exist in one copy of an area, then it can be destroyed and a new copy can be created from existing copies of the area, using an online utility. FPE does not stop the area (and make it unavailable to application programs) until the number of write errors in one program exceeds ten.

There are two online utilities to create and maintain a multiple ADS (Area Data Set) environment. DEDB Area Data Set Create Utility dynamically increases the number of ADS per area as an alternative to Data Base Recovery Utility, and provides an online migration tool when moving from one device type to another. It operates concurrently with online processing. DEDB Area Data Set Compare Utility is to check for inconsistencies between multiple ADS. It also operates concurrently with online processing.

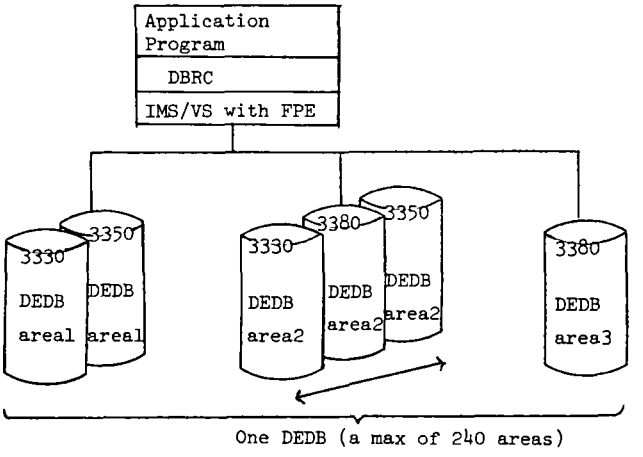


Fig.4 Multiple Area Data Sets in DEDB

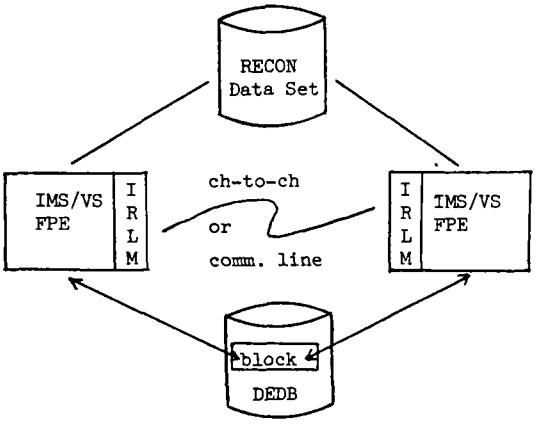


Fig.5 Data Sharing at the block level

9. Data Sharing

The data sharing capability which was implemented in IMS/VS Version 1 Release 2 has been expanded for DEDB as major function of FPE. There are two levels of data sharing: area level and block level.

At the area level, IMS/VS systems share an area of the DEDB. In this case, application programs in each IMS/VS system can read the data concurrently, but one IMS/VS system can update the data. Sharing is controlled by DBRC using RECON (Recovery Control) data set.

At the block level, multiple IMS/VS systems can update the data (different control intervals in the same DEDB) concurrently. Write requests are serialized and access to each block is controlled. The IMS/VS systems can be running in a maximum of two processors, connected channel-to-channel or via a communication

line using ACF/VTAM (a communication access method).

For block level sharing, both DBRC and IRLM (IMS Resource Lock Manager) are required. DBRC authorizes the use of the DEDB area by the IMS/VS systems sharing the data base. IRLM keeps data integrity during the access to a block by serializing application program requests for the block and ensuring that two programs do not access the same block for update at the same time. An enqueue table is associated with each IRLM. For a block to be accessed, IRLM creates a resource lock name (ID) consisting of the data base name, area name and relative byte address of the block. If it is not found in the enqueue table, the ID is placed into it, and, at the same time, sent to the IRLM in the other system. Upon receipt of an OK response, the access to the block from this IMS/VS system takes place.

The data sharing facility is to potentially increase processing capability and system availability by allowing multiple IMS/VS subsystem in multiple processors to share the same data bases.

10. Summary

The expanded Fast Path feature (FPE) consists of approximately 90K lines of code, 10K of which are written in PLS. It has over 440 modules/macros, one third of which have been borrowed without modification from the FP in IMS/VS 1.2.

FPE development has been an extremely complex project because of many changes and expansions to existing highly sophisticated system program structure and components and because of simultaneous development of major functions to be integrated into the base system (IMS/VS 1.3) whose development has been going on thousands of miles away across the Pacific.

The DEDB designed to enable high-speed processing with simplified hierarchical data base structure and with the use of VSAM improved control interval processing (ICIP) has been extended enough for banking and other on-line applications.

With the introduction of the area concept, a VLDB as large as 960 GB can be built. An area can be added or deleted dynamically. A failure in an area does not prevent the use of other areas. When an error occurs in the area data set, only the particular control interval (physical block) in error is deactivated (block deactivation) and other blocks can still be accessed.

With the integration of FPE into the IMS/VS base, IMS/VS is now emerging as a powerful high-performance, full-function DBMS even to those application areas where roll-your-own systems have been dominant.

Acknowledgement

As the manager of this project, I wish to express my hearty thanks to all the members of my team (M. Fukuda, A. Yamashita, P. Perner, et al) and other people involved in this project in the IBM Japan SDC for their devoted efforts and continued hard working as well as to a number of people of IBM General Products Division Santa Teresa Programming Center, San Jose, Calif. for technical and management support.

Database Management Systems for Very Large Scale Applications

Kenji Suzuki and Sajio Ikeda

Yokosuka Electrical Communication Laboratory
Nippon Telegraph and Telephone Public Corporation (NTT)
P.O. BOX 8, Yokosuka, Japan

Introduction

NTT is a public corporation which serves 40 million subscribers on its telephone network. Non-telephone services have also been offered to the public in recent years. NTT began data communication services in 1968. DIPS (Dendenkosha Information Processing System) has been developed for use as NTT's standard computer system. This paper offers a look at database management systems implemented on the DIPS computer used for very large scale applications.

Most of application systems on the NTT's data communication services are nationwide and have real-time database processing applications. Examples are the Motor Vehicle Registration System and Social Insurance System. Trends in database system scale for NTT's data communication services are shown in Fig. 1. As can be seen in this figure, the trends in database size and in transaction rate are toward a larger scale and heavier traffic.

For the construction of very large database systems, DEIMS (DEndenkosha Information Management System) and DORIS (Dendenkosha Online system for Retrieval of Information and Storage) were developed as DBMSs based on the CODASYL model [TAKA80].

Implementation for Very Large Databases

DEIMS is a general-purpose DBMS for real-time system use, and is suitable for very large scale and heavy traffic applications such as nation-wide systems or banking systems. DORIS is a DBMS for time-sharing system use, and is suitable for high-speed retrieval applications such as information retrieval systems. DEIMS and DORIS realization techniques are shown in Table 1. For DBMS realization, the entries shown in Table 2 have been improved in order to allow the construction of very large database systems.

To improve data independence, a multi-level database method is realized. This method improves physical data independence, in the manner shown in Fig. 2. The physical elements related to the performance and characteristics of application systems may be changed without affecting existing application programs and data structures.

To improve file efficiency, two techniques are used. One realizes the set concept based on the CODASYL model. A pointer chain method is used for the construction of network or tree structures, and a physical arrangement method is used to reduce the file size for pointers if the access paths for the records mostly conform with the tree structure.

The other technique is a data compression method for efficient storage of large amounts of data. In addition to the general methods of null data deletion and blank suppression, this method is realized by encoding character string data in variable and fixed codes according to the appearance frequencies of the characters. In a typical document database, this makes it possible to reduce the file size by 40%.

To improve process efficiency, effective buffer management and exclusive control methods are realized. There are two kinds of buffers: a system buffer for shared index pages and a user buffer for non-shared data pages, where pages are the basic unit of physical input or output. These buffers are managed by a page substitution method called the LRU or Biased LRU method. Concurrent accesses to the database is controlled by the DBMS. A page is the unit of exclusive control besides the area as specified by the CODASYL model. Deadlock is prevented by the DBMS.

Current Status

DEIMS services started in January 1979, and are used by 7 database systems. For example, the Motor Vehicle Registration System now has a database of 34 million automobiles that is being accessed by 81 offices throughout Japan. A distributed database management function based on DCNA (Data Communication Network Architecture) is under development [SUZU82]. The current version supports several enhanced functions.

(1) A database organization and access control method with a page split function is employed that is capable of efficient management of rapidly growing tree structure data such as in a banking system [SUZU81].
(2) A relational end-user language interface is implemented [HASH81].
(3) Japanese Kanji characters are accepted.

DORIS services started in December 1979. Information producers have so far used this system to make on-line information retrieval possible for documents, newspaper accounts, as well as economic and statistical databases. About one thousand user terminals are in use at present. Two non-procedural type query languages have been developed. One is for bibliographic category databases, and has retrieval functions for logic, weight and thesaurus retrieval. The other is for non-preplanned retrieval from general databases. It provided a relational interface. For Japanese data represented using the Kanji character set, a Kanji data attribute has been added to the database definition language specifications.

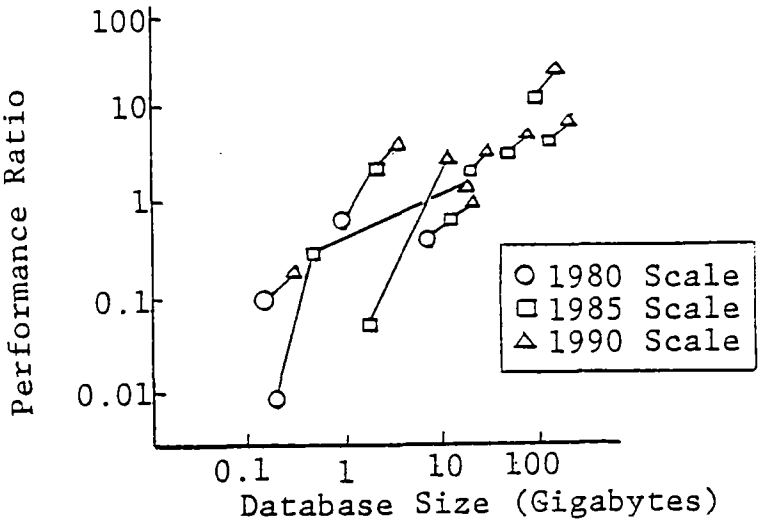


Figure 1 Projected trend in database system scale for NTT's data communication services.

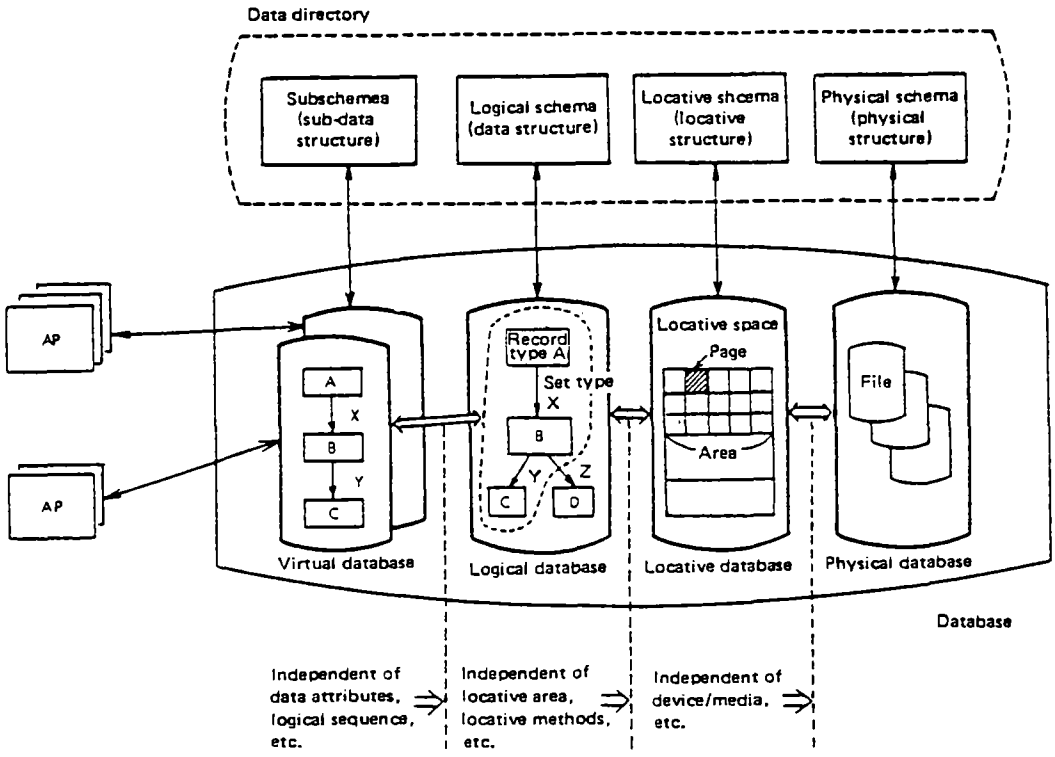


Figure 2 Data independence improvement with multi-level databases.

Table 1 Realization techniques.

Technique			DEIMS	DORIS
Database definition techniques	Data structures		Sequential structure Tree structure Network structure	Sequential structure Tree structure
	Data independence		Schema/subschema structure Multi-level schema structure	Schema/subschema structure
Database manipulation techniques	Host language interface		SYSL* COBOL	SYSL* COBOL, PL/1
	Non-procedural language		Relational query	Relational query Logical query
	Set realization		Pointer chain Physical arrangement	Physical arrangement
	Retrieval	Definite data value retrieval	Index sequential Direct Inverted index	Index sequential Inverted index
		Indefinite data value retrieval		Logic Weight Relational
		Data relationship retrieval	Set	Set
	Updates		Any time and immediate	Subsequent and all together (Batch)
	Data compression		-	Compression
Access control techniques	Exclusive control		Area unit Page unit	Database unit
	Buffer management		User buffer System buffer	User buffer
	Reliability		Journaling Rollback Duplicating	Rollback
Support techniques			Database creation Reorganization Statistics Recovery	Database creation Reorganization Statistics

* SYStem description Language developed for DIPS.

Table 2 Implementation of techniques oriented towards very large data volume and high performance.

Objective		Techniques	Implementation	DBMS
Data independence improvement		Database construction techniques	Multi-level database method	DEIMS
Performance improvement	File efficiency improvement	Set realization techniques	Physical arrangement method	DEIMS DORIS
			Pointer chain method	DEIMS
		Data compression techniques	Variable coded method	DORIS
	Process efficiency improvement	Buffer management techniques	System/user buffer method Biased LRU method	DEIMS
		Exclusive control techniques	Page unit exclusive control method	DEIMS
Reliability improvement		Fault processing techniques	Rollback recovery method	DEIMS

References

[HASH81] M. Hashimoto et al., "Time Sharing System Oriented Database Management System," Rev. E.C.L., NTT, Vol.29, Nos.1-2, pp.16-31, 1981.

[SUZU81] K. Suzuki et al., "A Database Control Function for Situations Where Data May Increase Rapidly," Rev. E.C.L., NTT, Vol.29, Nos.1-2, pp.1-15, 1981.

[SUZU82] K. Suzuki et al., "Implementation of a Distributed Database Management System for Very Large Real-time Applications," COMPCON Fall, pp.569-577, 1982.

[TAKA80] S. Takahira et al., "Very Large Data Amount and High Performance Oriented Database Management Systems," Rev. E.C.L., NTT, Vol.28, Nos.3-4, pp.229-245, 1980.

The Data Communication System for Nationwide Banking System

Haruhiko Imamura

NTT Data Communication Bureau

17 Mori Bldg., 1-26-5 Toranomon,

Minato-ku, Tokyo, 105 Japan

The Data Communication System for Nationwide Banking Activities in Japan was put into service in April, 1973, covering 87 banks as well as the Bank of Japan. Most major banks which can have their own networks are not included in the system. As the number of transactions processed by the system continued to increase, a new system had to be developed which had processing capacity of 5 times the previous system. It was put into operation in February, 1979. This system is capable of handling 3.4 million exchange messages per day. The network covers all private banks and their branch offices throughout the country.

The number of participating banks increased from 87 banks and their some 8,000 branch offices to 701 banks and 19,500 branch offices. Data traffic is expected to reach 3,400,000 messages daily by 1986.

1. System Configuration

The basic system configuration is shown in Fig. 1. This system consists of a central computer center, terminal equipments and communication lines necessary for connecting all the facilities. The central computer center is located in Otemachi Telephone Office, Tokyo. It consists of 2 on-line systems and one off-line system (which can also be used as an on-line system) for high capacity and reliability. These systems use Dendenkosha Information Processing System (DIPS)-11, Model-30, and a single processor system at the beginning of service. Each system can be upgraded to a tightly coupled multiprocessor system for higher capacity and reliability.

2. System Features

In this system, as shown in Fig. 1, more than two terminal equipments of the same category are provided at each bank. Each terminal equipment is connected to one of the two on-line systems in the central computer center. Such a connecting method has the following merits.

(a) Because the load is balanced between the two on-line systems, each can furnish maximum processing capacity.

(b) If failure occurs in either system at the central computer center, the other CPU is able to continue its operation. In the meantime, the failed CPU can be replaced by the CPU of the off-line system.

(c) In addition, if trouble occurs in either the terminal equipment or the communication circuits, it is possible to send messages via a second route.

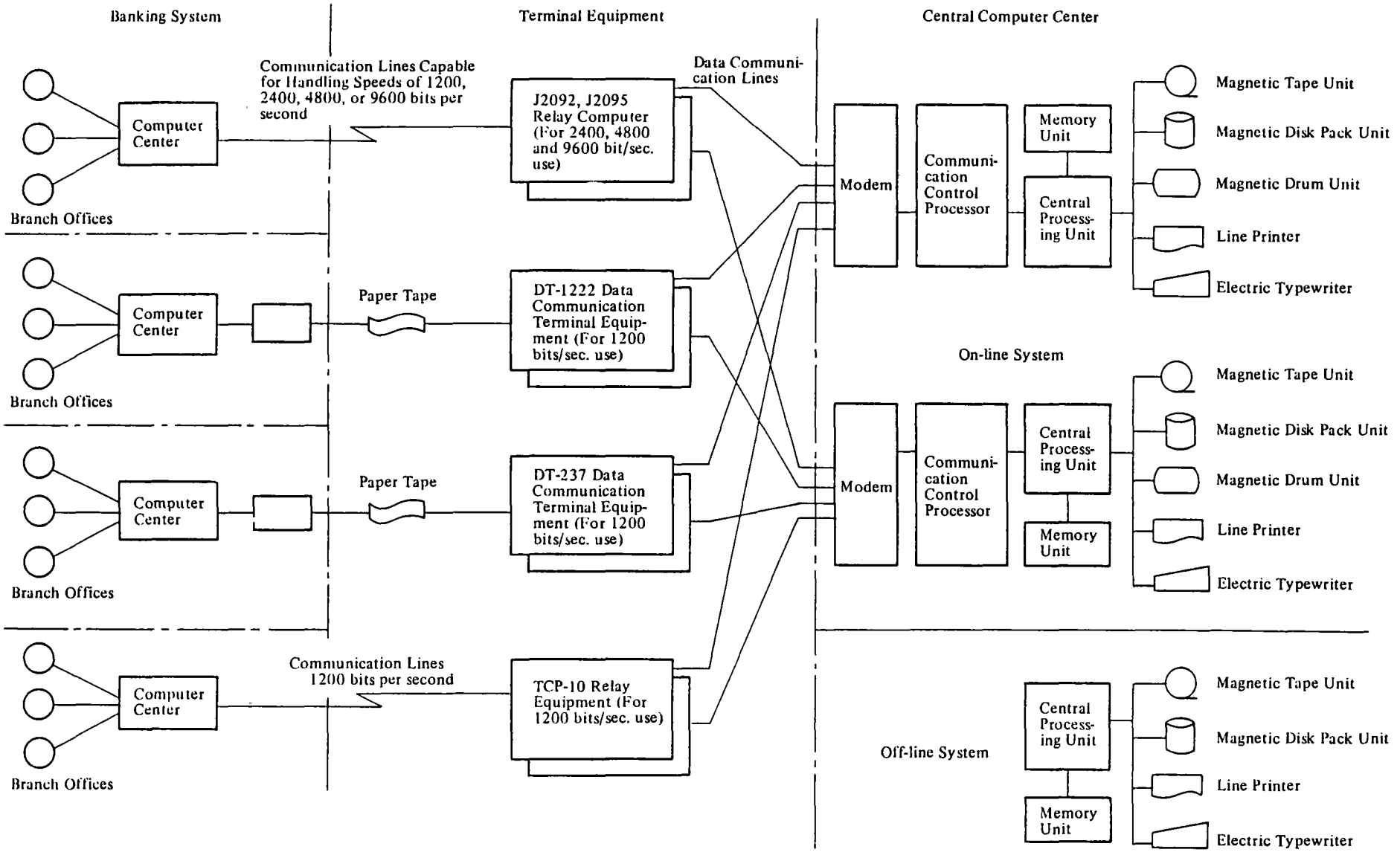


Fig. 1 System Organization

(d) A Relay Computer (J2092, J2095 in Fig. 1) has also been developed, which is more compact and provides higher performance than the previous one (J2015). It can process 44 thousand messages per hour. And a new relay equipment (TCP-10 in Fig. 1), which has similar functions as the relay computer and can process 2,000 messages, is being developed. The DT-1222 data communication terminal equipment will be replaced by the TCP-10 from September, 1983.

(e) For communication circuits, 9,600 bits/second, 4,800 bits/second and 2,400 bits/second lines are used between the Central Center and the Relay Computers. High Level Data Link Control procedure was adopted for transmission control.

3. System Usage

The system usage status for the first six months of 1979 is shown in Table I. Table II shows expected transaction rates.

Table I

Items \ Month	February	March	April	May	June	July	Total
Total Number of Messages for the Month (10,000 Messages)	889	1,484	1,382	1,582	1,566	1,616	8,499
Number of Messages during Peak Day (10,000 Messages)	133	101	102	122	105	127	—
Total Operation Time for the Month (Hours)	192	338	313	332	341	338	1,854

Table II

Items \ Year	1979	1983	1986
Number of Messages during Peak Day (Thousand Messages)	1,400	2,300	3,400
Number of Messages during Peak Hour (Thousand Messages)	320	530	790
Average Message Size (Characters)	From center to terminal: 178 From terminal to center: 158		
Central Processing Unit Configuration	Single-Processor	Multiprocessor	
Processing Performance (Thousand Messages)	500	900	

CADETT: Computer Aided Design and Engineering Tool for Toyota

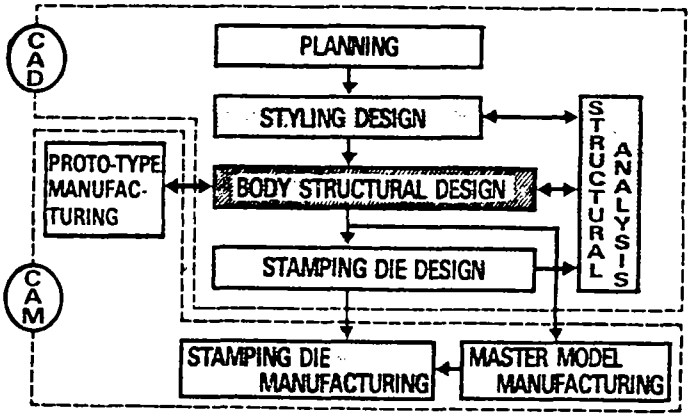
Yukio Sakai
Body Engineering Department

Yasuhiko Kuranaga
Information Systems Department-I

Toyota Motor Corporation
1-Toyota, Aichi Pref., Japan 471

1. Introduction

The design and engineering work on car body structures requires a high level of human intelligence, and for many years it did not seem amenable to computerization. However, recent advances in computer graphics have made interactive use of computer systems possible for such applications. Under emerging needs for new products, we at Toyota Motor Corporation have developed and implemented a large-scale computer graphics system for the 'body structural design' process. As shown in Fig. 1, the process is a major step in a car body development, from planning to the manufacturing of stamping dies.



〈FIG. 1〉 PROCESS OF CAR BODY DEVELOPMENT
(CADETT IS APPLIED TO BODY STRUCTURAL DESIGN)

The system, named CADETT (Computer Aided Design and Engineering Tool for Toyota), has many helpful functions for car body engineers. They may be classified into two categories:

1. Design and engineering aids, which enable fast and accurate design and drafting.

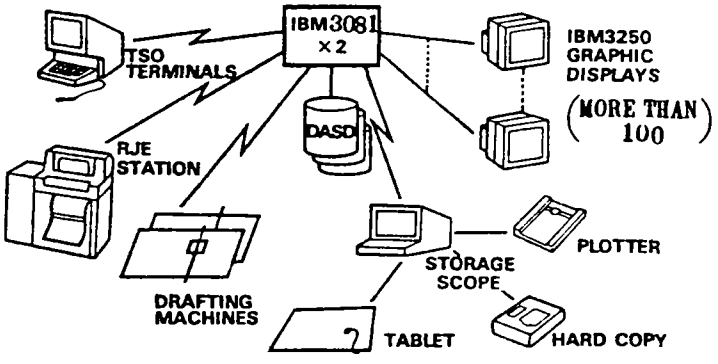
2. System management aids, which reduce the workload on DP department personnel.

CADETT is widely in use for the design and drafting of car body shell, inner trim, garnish, etc., and it has contributed to substantially improved productivity and quality of car body design and engineering work. This paper describes the CADETT system as an example of a working CAD system which makes use of IBM's IMS (Informa-

tion Management System) and a relational database system called RIX (Relational Interface Extension).

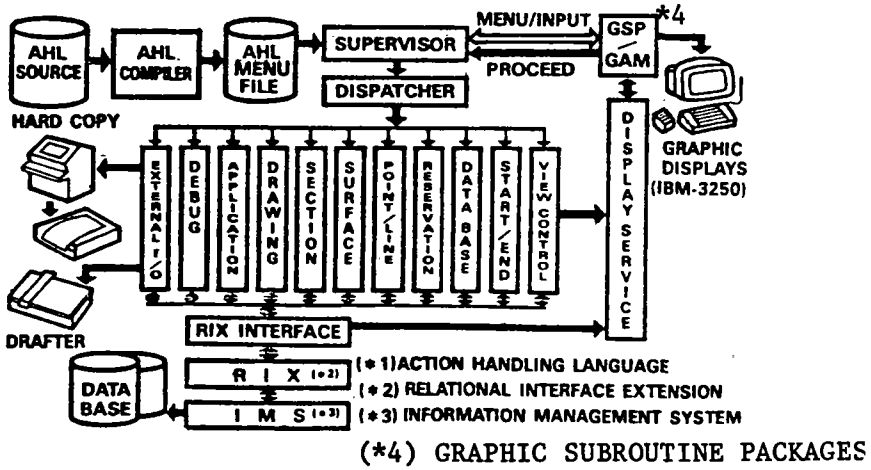
2. Outline of the System

Fig. 2 shows the hardware configuration of the CADETT system. Host computers are two IBM 3081's. Many IBM 3250 graphic displays are attached to each host for use by car body designers and engineers. There are two drafting machines for producing hard-copy drawings to check against the designs done on the screen and to communicate information to users who do not have a CAD system.



〈FIG.2〉 HARDWARE CONFIGURATION OF CADETT

The software structure of the CADETT system is shown in Fig. 3. CADETT has been implemented largely in PL/I, and it consists of about 1500 program modules and 300,000 instructions.



(*4) GRAPHIC SUBROUTINE PACKAGES

〈FIG.3〉 SOFTWARE CONFIGURATION OF CADETT

The software structure of CADETT may be broadly classified into four areas.

1. User Interface: One user interface is the graphic displays from which the users communicate with the Supervisor of the system. The Display Service function provides a means of outputting the database contents and the result of computations. The users can make full use of the functions of the graphic displays to design, modify and store their drawings. Another interface is the hard-copy units.

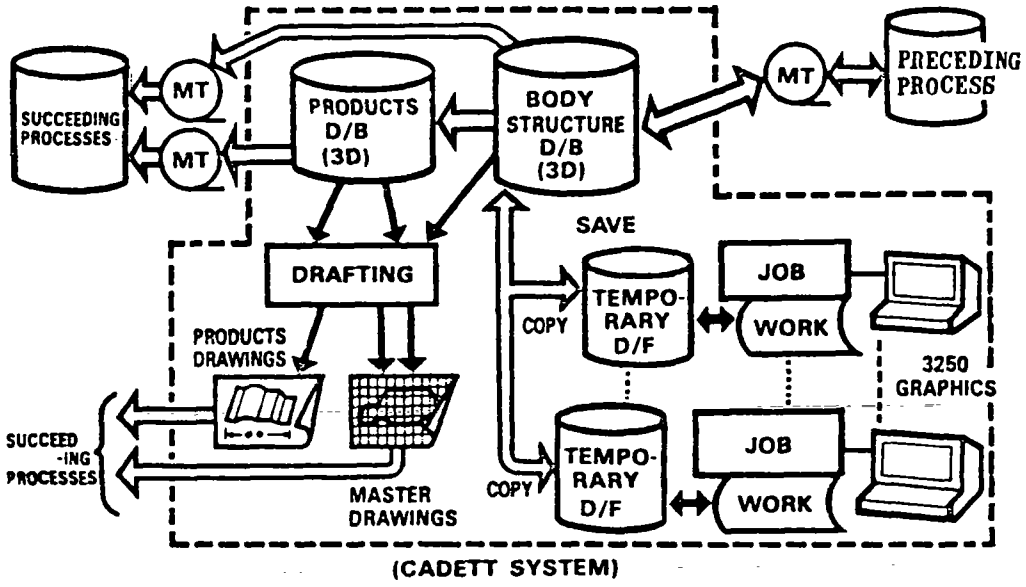
2. Functions: The functions of the system are indicated in the center of Fig. 3. It is easy to add or delete functional modules and to maintain them. All programs are classified into some nonoverlapping functional groups.

3. Menus: The Action Handling Language (AHL) has been implemented to process menus that provide the users with a friendly interface to the functions of the system. The separation of menu processing from the application functions has not only reduced the system development time but also, perhaps more importantly, made menu processing completely independent of the computation algorithms used in the application functions.

4. Database Systems: At the bottom of Fig. 3 is shown what may be the most important component of the CADETT system. The intermediate and final drawings are stored in the master database. Two types of software are needed. One is for storing and retrieving the drawings. The other is for providing an easy means of modifying the drawings, since the designers create and change their designs often. IMS is used for the former function, and RIX for the latter.

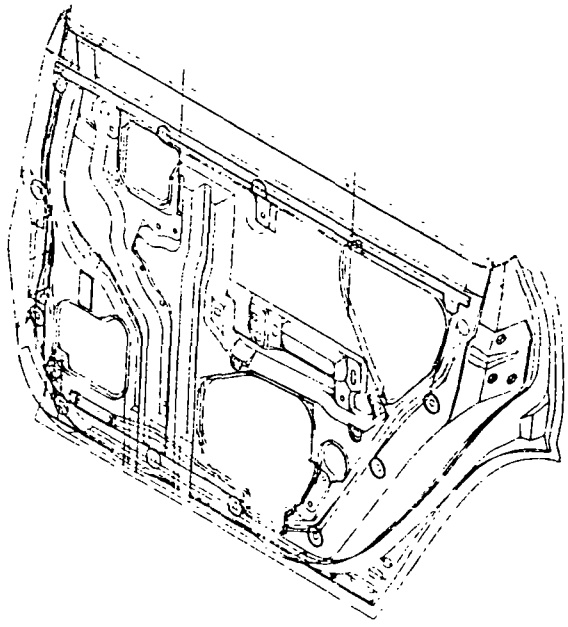
3. Database Systems

Now we will describe the flow of data through the system. Fig. 4 illustrates the data flow from a preceding process, through CADETT, to a succeeding process. The preceding process is the 'styling design' process of Fig. 1. The styling data is transferred to the CADETT system by use of magnetic tapes(MT). The data is stored in the Body Structure Database. Body engineers copy the original styling data from this database to their own temporary data files(D/F), and do their design and engineering work on assigned portions of the car body. Several engineers may work on the same drawings. Manipulation of data in the temporary files is done by RIX.



⟨FIG.4⟩ DATA BASE IMAGE OF CADETT SYSTEM

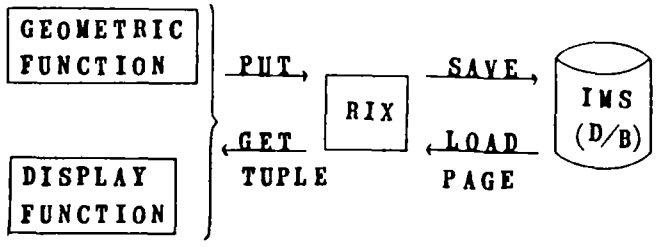
The result of body engineering, that is, the final Body Structure Database that results from the engineering work, is rather complex and its hard copy, such as a master drawing, is in effect an X-ray picture of the car body. Fig. 5 shows an example of such a drawing, which was designed and drafted using CADETT. Note that all data are 3-dimensional.



〈FIG. 5〉 REAR DOOR INNER PANEL

Once the design is completed for the whole car, the Products Database is created containing data for each part of the car. Master drawings may be obtained using data from either database, depending on the needs. In the later stages of car body development where drawings are the key data to work with, drafting of these drawings is very important. For other processes such as NC(numerically controlled) cutting of master-model and stamping dies making where the process is computerized, the drawings data are transferred by means of magnetic tapes for loading on the NC cutters.

IMS and RIX are used to manage the drawings data and geometrical data in the database. The relationship between IMS and RIX is shown in Fig. 6.



〈FIG. 6〉 RELATIONSHIP BETWEEN IMS & RIX

IMS is used to store and retrieve data from the database in 4K-byte pages. In particular, the Body Structure Database is managed by IMS. IMS was chosen because of its proven capability as a database management system. Its file-integrity features are very helpful to CADETT.

RIX is a database processing software developed by IBM Japan, and is only available in Japan. It provides the users with work areas which contain drawings retrieved from the IMS database. RIX presents a tuple(record)-level interface to CADETT's geometric function and display function. It transforms a final drawing into a simple tree structure and files it into the IMS database.

Concluding Remarks

Ever since it became operational in 1979, the CADETT system has served to increase productivity in car body design work. We credit much of its success to the use of IMS and RIX and the friendly, conversational interface to the functions of the system. We still have much work to do in refining the system and extending it to other applications areas. Our plans call for extending CADETT to the design of the engine, chassis, and other major automotive parts.

THE TRAVEL RESERVATION ON-LINE NETWORK SYSTEM

Koichi Tsukigi*
Yohtaro Hasegawa
Japanese National Railways
Chiyoda-ku, Tokyo 03(212)6311

1. Introduction

Present-day railroads face competition from other forms of transportation. The development of airplanes and automobiles has significantly impacted passenger transportation by railroads. But railroad transportation is more economical and the primary means of passenger transportation in Japan is still by railroads, since cities are within easy distance of each other.

The development of a train seat reservation system by JNR named MARS has been sustained by the growth in passenger transportation operations of JNR and the progress in computer technology in Japan.

In 1980 , the on-line linkage of MARS with computer systems of major travel companies (Japan Travel Bureau Inc. (JTB),Nippon Travel Agency Co.,Ltd. (NTA), and Kinki Nippon Tourist Co.,Ltd. (KNT)) was completed. Recently, one more system (Tokyu Tourist Co.,Ltd. (TTC)) has joined the network. The systems of these travel companies are also on-line computer systems , mainly for selling tourist commodities such as hotel accommodations. This computer network is very large in scale and unrivaled in the variety of information processed. Since domestic tours are sold as a commodity by each system, it offers a great help to travellers.

2. Outline of MARS

Before the computerized reservation system was introduced , booking of JNR trains had been handled manually. The first experimental system for seat reservation was operational in 1960, when computer technology was rudimentary. This system was called MARS 1(Magnetic electric Automatic Reservation System). Since then , to cope with the rapid growth in the number of passenger seats resulting from expansion of the Shinkansen (bullet trains), etc. , the MARS system was replaced and improved step by step, expanding its services.

Presently MARS 105 handles 1,000,000 seat reservations per day. About 1,800 specially-designed terminals have been installed at 500 JNR stations and 450 travel company offices spread all over Japan.

Two more on-line systems for seat reservations were also developed. One is a telephone reservation system , called MARS 150 , serving Tokyo area. The other is a group and party reservation system, called MARS 202. Both MARS 150 and 202 are connected to MARS 105. These three systems provide fast and convenient traveller services , and together they make up an integrated sales management system.

* Author's current address : Railway Technical Reserch Institute , JNR , Kokubunji-shi , Tokyo 0425(72)2151

2.1 MARS 105

MARS 105 has adopted a tandem configuration to process high volumes of transactions , more than 100 calls per second. As Fig.1 shows , MARS 105 consists of a front-end system, the Communication Computer subsystem (CC), and a back-end system, the File Computer subsystem (FC).

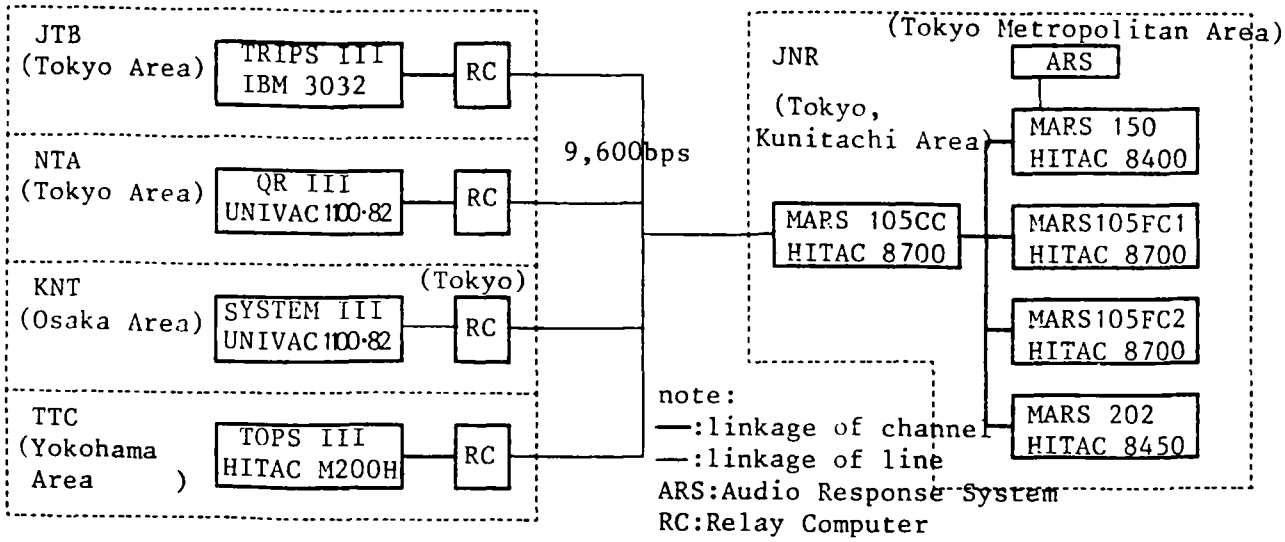


Fig.1 Configuration of the travel reservation network system

The CC controls the communication network , the terminal status and traffic flow to the FC. The CC itself is a tightly coupled multiprocessor system and one operating system controls two processors that share a common main memory.

MARS 105 FC is composed of two single-processor systems , FC 1 and FC 2. The FCs control the train and seat information files and process the main on-line applications for reservations. Both the CC and FCs operate under a specially-designed operating system (RTCS , Real-Time Control System) and file management system to obtain high performance and reliability.

2.2 MARS 150

The telephone reservation system , MARS 150, made it possible for customers to reserve Shinkansen seats by use of touch-tone telephones. The inquiry data is keyed in through a touch-tone telephone and the response is giving in human voice by an audio response system. After that, tickets for the reserved seats are issued from a terminal of MARS 105 using the reservation number included in the voice response.

2.3 MARS 202

MARS 202 was developed for group and party passenger travellers. This system provides information on seat availability over long periods and can reserve up to 980 persons in one operation. Additionally , it can handle complete package tours , including hotel accommodations, cars, meals, etc. to meet the growing demand for fast and easy holiday reservations. These functions were improved by the system linking with travel companies.

3. On-line linkage of MARS with the travel companies' systems

The travel reservation on-line network system links MARS with each of the sales systems of the four major travel companies in Japan. The names of the travel companies' systems are "TRIPS III" for JTB , "QR III (Quick Response)" for NTA , "SYSTEM III" for KNT and "TOPS III" for TTC.

3.1 Purposes of linking the systems

(i) Expansion of the sales networks in the market

MARS has about 1,800 units of terminal equipment for reservations , while JTB, NTA, KNT and TTC have about 900, 200, 150 and 150, respectively. By linking these systems , JNR's ticket selling network is expected to have more than 3,000 terminal units.

On the other hand , 200 units connected to MARS 202 are available to reserve tourist commodities in each travel company's system. As a result, the on-line network system contributes to the expansion of both networks in the market.

(ii) Improved efficiency in selling integrated tour packages

Selling of train tickets alone is not sufficient to meet the increasing demands for tourist commodities. If a variety of tour packages are sold , whose values are enhanced by such means as combining coupons for hotel accommodations , ship and bus reservations , etc. with JNR tickets , it would greatly increase JNR's marketing capability.

MARS and the travel companies' systems provide facilities to efficiently assemble the tourist commodities filed in their systems into a variety of integrated tour packages.

(iii) Improved efficiency in settling accounts

In the past , JNR's tickets were mostly sold manually by travel companies. The companies spent a considerable amount of labor on account settlement operations , because the number of JNR's tickets comprised about 70 percent of all that the companies handled and the volume of sales was very large. The network system attempts to computerize these operations and improve efficiency in settling accounts.

3.2 Operations carried out with the linkage

(i) Ordinary selling operation

The selling of JNR's tickets through each travel company's system is called an ordinary selling operation. The company's system is regarded by MARS as being identical to its own terminal equipment and MARS processes booking and cancellation of reserved seats. In addition , it is possible for MARS to sell tourist commodities such as hotel accommodations in the companies' systems.

(ii) Selling operation of planned packages

A planned package means a tourist commodity which is set in advance as a model

course. Various tickets required for the tour, such as reserved seat tickets, fare tickets , hotel accommodation coupons and sightseeing coupons , are combined in one set and sold as an integrated package.

The network system has made it possible to secure necessary commodities from other systems on an on-line basis.

(iii) Information transmitting and checking operation

Information transmitting operation means to transmit data necessary for daily operations to other systems on an on-line basis. These data include marketing data, inspection settlement data and statistical data. As each system sells various commodities in other systems through the network , the checking of daily sales volume is one of the most important jobs. At the end of each working day, it confirms the number of tickets and the revenue.

3.3 Network configuration

The configuration of this network system is illustrated in Fig.1. MARS is linked with each travel company's system via data relay computers by 9,600 bps communication lines. For data communication control procedure , HDLC (High level Data Link Control)-ARM was adopted. At the time JNR adopted HDLC , standardization work of ISO had not yet been completed. Therefore, JNR worked out the details of the procedure and added some new functions for auditing line conditions and daily operations.

3.4 Message flow control

This network system is a typical distributed processing system and it is important for the system to control messages flowing in the network. The control procedure was developed on the basis of the PST (PSeudo Terminal) method which has been adopted by the linkage with subsystems of MARS.

The characteristics of flow control are as follows.

(i) Mutal independence of the operations peculiar to each system

Any trouble occurring in the network has no effect on the operations which are not related to the linkage in question.

(ii) Securing uniqueness of messages

The control procedure prevents messages from being lost and screens duplicate messages.

(iii) Control of flow quantity in the network

At peak time , MARS makes it possible to decrease the flow quantity in the network.

(iv) Simple and quick recovery

The recovery procedure is simplified by using the PST method. The system in which some trouble occurred , may hasten to execute necessary and minimum

recovery procedure by itself. Then the whole system returns to normal by gathering the recovery information transmitted from terminals and PSTs in other systems.

3.5 Distributed data processing

MARS operates under a specially-designed file management system to obtain high performance and each company's system uses DBMS developed by its computer maker to shorten the period for its development. The facilities of each local system are incorporated into the over-all network as distributed functions.

In performing the ordinary selling operation , for example , MARS processes booking of train seats and calculation of charge , while each company's system translates code information into printing images of tickets and issues tickets for the reserved seats.

The selling operation of a planned package requires sophisticated and distributed processing through the network. In this case, some course files for planned packages are prepared in advance , containing indices of accommodations necessary for these packages. Fig.2 shows that a course file is divided into parent and child parts related to commodities of each system. Every time a request for selling the package enters the system , the commodities indicated by each part of the course file are secured separately in each system. Additionally , two more types of distributed processing for the operation are provided for efficiency on selling.

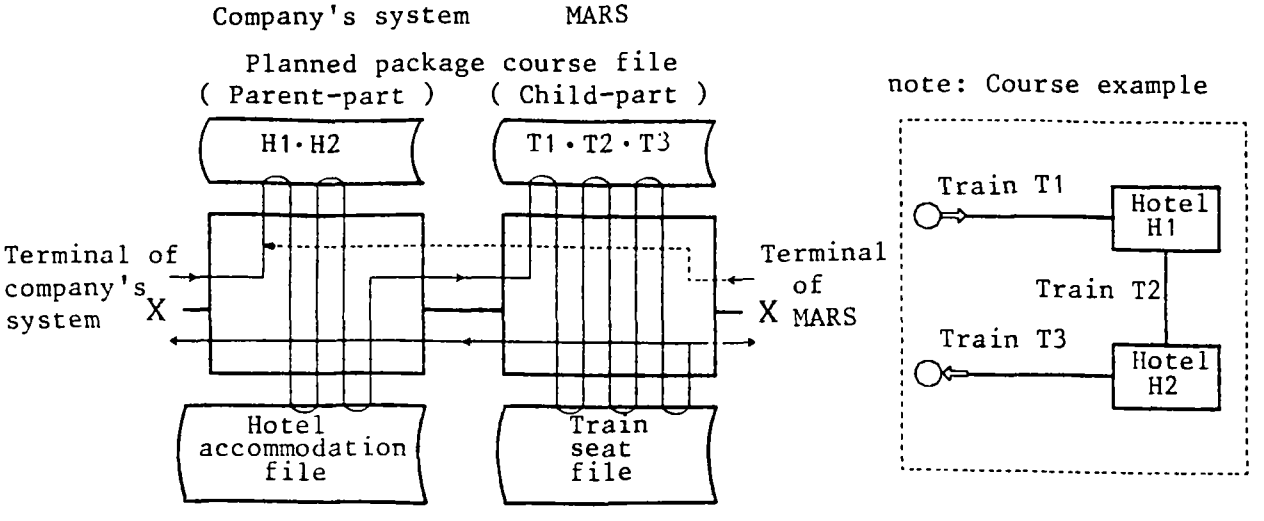


Fig.2 Example of a distributed processing for selling a planned package

4. Summary

During the development of the on-line network system , it took a long time to decide common specifications of the system and debug its software , because MARS and the companies' systems had their own design targets. But nowadays the system is operating very satisfactorily and the peak traffic volume through the network totals more than 250,000 transactions per day.

Heterogeneous Distributed Database System: JDDBS

Makoto Takizawa
Japan Information Processing
Development Center (JIPDEC)
3-5-8, Shibakoen, Minato-ku,
Tokyo 105, Japan

Introduction

A distributed database system (DDBS) is a system which is composed of various database systems (DBS's) with semantically related data connected by communication networks, providing users with one logical DBS service. SDD-1 [ROTH80] and Polypheme [ADIB80] are DDBS's composed of homogeneous relational DBS's. JDDBS (JIPDEC DDBS) [TAKI78-82], which has been under development since 1977, attempts to integrate existing heterogeneous DBS's. In order to realize a DDBS, the heterogeneity and distribution problems of the DBS's have to be solved. Our approach to solving the heterogeneity problem is called the four-schema structure [TAKI78,79][Fig.1]. Recently, [DAYA82, DEVO82] have proposed similar concepts. By defining a common model view called a local conceptual schema (LCS) over each local internal schema (LIS), i.e. existing DBS schema, the heterogeneity problem is solved; this is called homogenization. The mapping information between both schemas, called heterogeneity information (HI), is generated and maintained at local sites. We adopt the relational model [CODD 70] as the common model because of its simplicity and closure property.

The distribution problem is solved by defining a global relational view called a global conceptual schema (GCS) over these local views (LCS's); this is called integration. In it, the mapping information between the GCS and the LCS's, called distribution information (DI), is generated. It is maintained in every site. At this level, the DDBS can be seen as one logical DBS. The EXS is an external schema defined over the GCS.

System Overview

JDDBS is composed of data modules $DM_1,\ldots,DM_n (n\geq 2)$ connected by a communication network (CN)[Fig.2]. A data module (DM) is composed of six submodules; database system(DBS); local database processor(LDP); heterogeneity information(HI); relational working storage(RWS); global database processor(GDP); distribution information(DI). The DBS is an existing database system. The LDP is a relational interface system over the local DBS, which translates the LIS into the relational schema (LCS) and generates the HI. It also translates a relational query into executable operations, e.g. COBOL DML's, makes the CODASYL DBS execute them, and returns the result as a relation in the RWS.

The GDP generates the DI from a GCS definition, decomposes a GCS query into queries referencing LCS relations and executes them in cooperation with other GDP's through the CN. For performance reasons, every DM has a complete copy of the DI.

The RWS is a file for storing relations, which plays the role of an interface between LDP and GDP as well as that of a working file of the GDP. The GDP performs relational operations on the relations in the RWS and transmits/receives relations to/from the RWS.

The CN provides highly reliable communication among GDP's in either one-to-one or one-to-many (broadcast) mode. In JDDBS, the distributed query processing based on broadcast communication is being implemented.

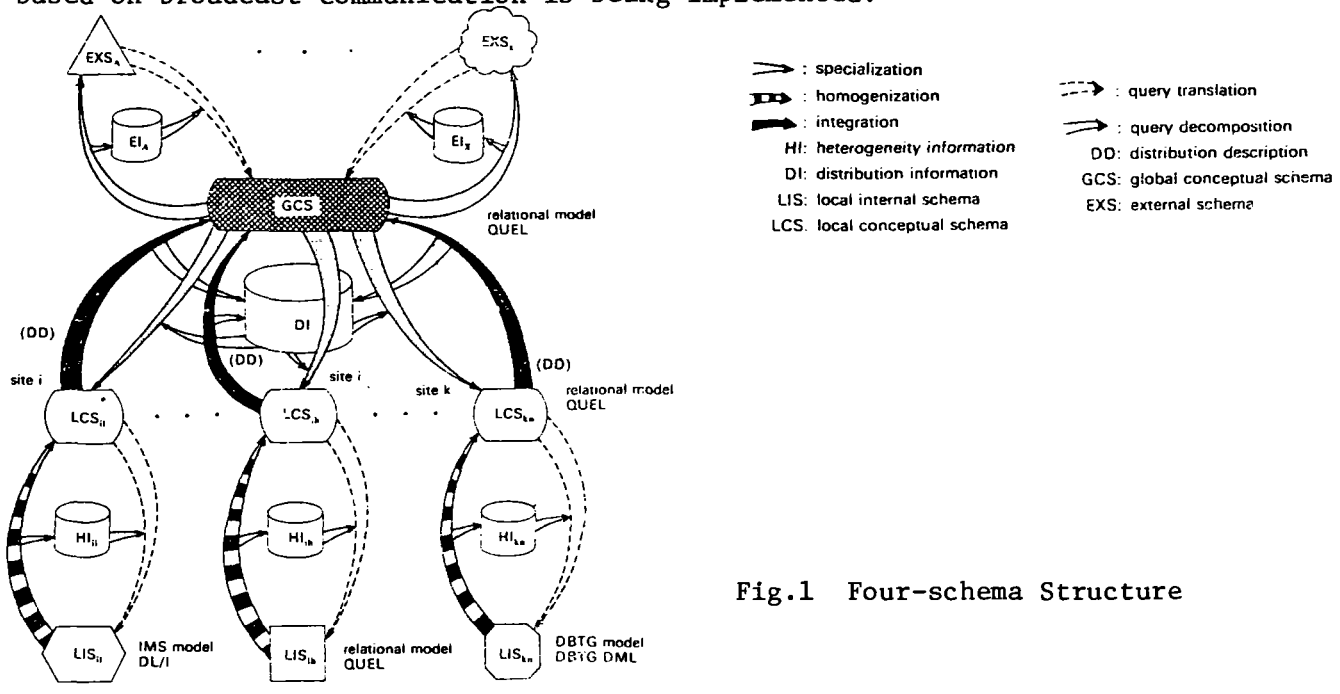


Fig.1 Four-schema Structure

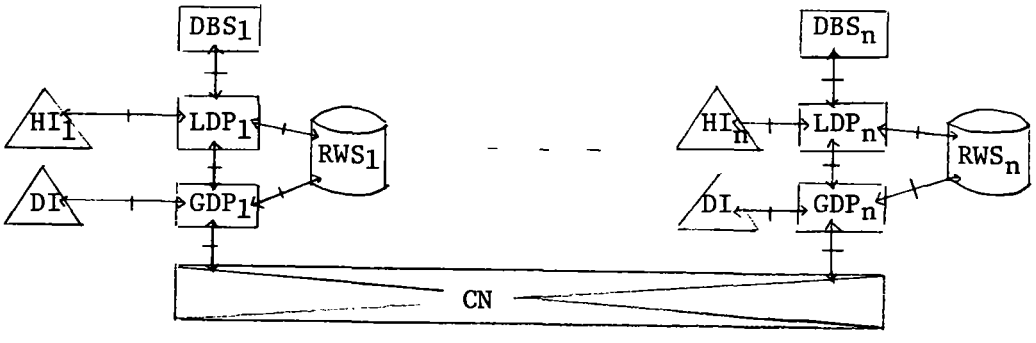


Fig. 2 JDDBS Overview

Local Database Processor (LDP)

The LDP is a common relational interface system over the local DBS. At present a CODASYL DBS is used. The LDP can translate a CODASYL schema (LIS) to a relational schema (LCS); this is called a schema translation. Also, the LDP can translate a relational query and update on the LCS into a COBOL DML program(this is called query translation[TAKI80]), make the underlying CODASYL DBS execute it, and return the result as a relation in the RWS.

Fig.3 shows a CODASYL schema representing organizations, their members and reports (ERL is a record type for representing relations between EMP and REP, which we call a link type). The LDP translates it into an LCS as shown in Fig.4 by converting record, set, and link types to E-, B-, and G-type relations, respectively. Underlined attributes are the primary keys and attributes marked @, called primary attributes, form the db-keys.

Relational LCS queries and COBOL DML programs differ in data structures (relational vs. CODASYL) and access units (set-at-a-time vs. record-at-a-time). For query translation, these two problems have to be solved. We have introduced a non-procedural query on the CODASYL data structure. By describing the LCS query

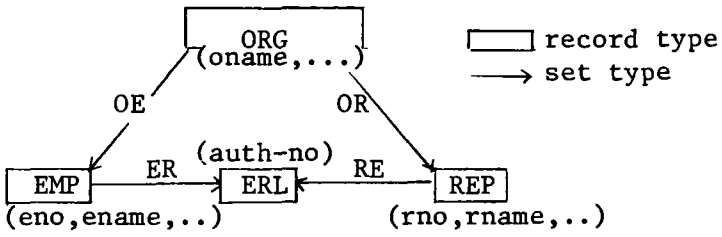


Fig. 3 LIS

E-type
ORG(@O,ONAME,...)
EMP(@E,eno,ename,...)
REP(@R,rno,rname,...)

B-type
OE (@E,@O)
OR (@R,@O)

G-type
ERL(@E,@R,auth-no)

Fig. 4 LCS of Fig.3

with this CODASYL query, the difference in data structures is removed. This is called structure transformation. For example, an LCS query against the LCS as shown in Fig.4, "find members of JIPDEC and their reports," can be written in QUEL [STON76] as follows;

```
range (e,EMP)(o,ORG)(r,REP)(er,ERL)(oe,OE)(or,OR);
get into R(r.rno, e.ename) where e.@E=er.@E and er.@R=r.@R and r.@R=or.@R and
  or.@O=o.@O and o.@O=oe.@O and oe.@E=e.@E and o.oname="JIPDEC";  --- (1)
```

In the CODASYL query, a join on the primary attributes is described by a predicate representing the owner-member relationship. For example, (e.@E=oe.@E and oe.@O=o.@O) is replaced by a predicate OE(o,e) where o and e are variables ranging over record types ORG and EMP, respectively. The CODASYL query can be described by a graph called a CODASYL query graph (CQG) whose nodes represent variables and directed edges the predicates representing set types. The symbols → and —| stand for target attributes and restrictions, respectively. Fig. 5 shows the CQG translated from query (1) above.

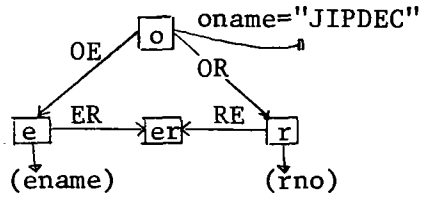


Fig. 5 CQG

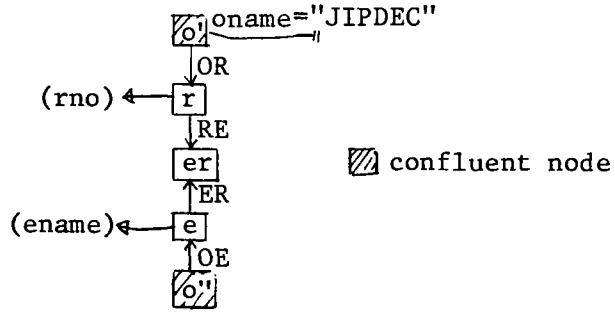


Fig. 6 Access Tree (AT)

An access tree (AT) representing an access path is generated by searching all CQG edges uniquely in depth-first order so as to minimize the number of record occurrences accessed. The number of record occurrences is estimated by statistics information of the underlying database, i.e. selectivities of items and connectivities of set types. The statistics are dynamically tuned up by the really accessed result of each program. Fig.6 shows an AT generated from the CQG of Fig.5. Since a CQG may include cycles, more than one AT node may be generated for a CQG node. Such AT nodes are called confluent nodes. In Fig.6, o' and o" are confluent nodes for a CQG node o of Fig.5. In our tree, all confluent nodes for a CQG node are included in a subtree whose root is one of them. They also imply identity predicates, e.g. o'=o" in Fig.6. In our tree, every confluent node can check the identity by comparing a record occurrence which the subtree root represents with record occurrences of the node.

By searching in preorder the branches and nodes of an AT, representing set types and record types, respectively, and generating COBOL DML's corresponding to them, e.g. find first...within..., find owner within ..., a COBOL DML program is generated. The program has to be executed by the CODASYL DBS. The LDP sends the JCL for compiling, linking, and executing the program using a network protocol or the Fujitsu's IPF facility.

The LDP has been operational on Fujitsu's AIM on the M-170F computer and Nippon Electric Co's ADBS on the Acos-700 computer since 1980. The current version of the LDP can process updates[TAKI82b], aggregates, and joins. The LDP on M-170F takes less than one minute to generate COBOL DML programs of a few thousands steps from LCS queries of only several lines. The LDP is coded in PL/I.

Global Database Processor (GDP)

The GDP not only generates the DI from a GCS definition over the LCS's but also executes a GCS query issued by the user. It cooperates with other GDP's through the communication network (CN). A global conceptual schema (GCS) is defined as a view over the LCS relations. This definition is stored in the DI.

A query on the GCS relations is translated into one referencing only the LCS relations by query modification [STON76]. Then the query is decomposed into LCS queries, each of which references relations in one LCS, and the decomposed queries are executed by the LDP's. This is called initial local query processing (ILQP). We note that the ILQP derives data as a relation from heterogeneous data structures.

Then the query is processed in cooperation with other GDP's. This is called distributed query processing (DQP). Our DQP is based on the broadcast communication provided by a local area network like the Ethernet. Let us consider a query consisting of equi-joins of relations R1,...,Rn at different DM's on attributes $a_1,\ldots,a_m$. In our algorithm, first one projection $Ri[a_k](=Ri')$, called a source, is broadcast from DM_i, and every DM_j receives it and joins it with Rj ($Rj \leftarrow Rj[a_k=a_k]Ri'$). DM_j broadcasts an acknowledgement (ACK_j) piggybacking a bit-map BM_j of $Rj[a_k]$ on Ri' and status information, i.e. cardinality of other join attributes, and waits for ACK's from all other DM's. On receipt of ACK_h with BM_h, $BM_j \leftarrow BM_j \wedge BM_h$. If all ACK's are received, R_j is reduced based on BM_j. A similar algorithm has been proposed independently by [KAMB82]. $R_p[a_q]$ whose size is the minimum is selected as the next source based on the status information from every DM. That is, the DQP is controlled in a completely distributed and dynamic manner. After applying this procedure to every join attribute, the result relations are sent to the output DM.

The GDP is under implementation and will be operational in Feb. 1983. We also plan to aplly our JDDBS to an office information system (OIS).

Acknowledgement

We would like to thank Dr. Won Kim for his helpful and instructive comments of this paper.

References

[ADIB80] Adiba, M. et al.: An Overview of the Polypheme Distributed Database Management System, Proc. of the IFIP, (1980), pp.475-479.

[CODD70] Codd, E. F.: A Relational Model of Data for Large Shared Data Bank, CACM, 13(1970), pp.337-387.

[DAYA82] Dayal, U. and Hwang, H. Y.: View Definition and Generalization for Database Integration in Multibase: A System for Heterogeneous Distributed Database. Proc. Berkeley Workshop on Distributed Data Management and Computer Networks, (1982), pp.203-238.

[DEVO82] Devor, C. et al.: The Design of DDTS: A Testbed for Reliable Distributed Database Management, Proc. 2nd Symp. on Reliability of Distributed Software and Database Systems, (1982), pp.150-162.

[KAMB82] Kambayashi, Y.: Query Processing in Distributed Database Systems, The IBM MFCS Symp., (1982).

[ROTH80] Rothnie, J. et al.: Introduction to a System for Distributed Databases: SDD-1, ACM TODS, 5(1980), pp.1-17.

[STON76] Stonebraker, M. et al.: The Design and Implementation of INGRES, ACM TODS, 1(1976), pp.189-222.

[TAKI78] Takizawa, M. et al.: Resource Integration and Data Sharing on Heterogeneous Resource Sharing System, Proc. ICCC, (1978), pp.253-258.

[TAKI79] Takizawa, M. et al.: The Four-schema Concept as the Gross Architecture of Distributed Database and Heterogeneity Problems, Jounal of Information Processing(JIP)(IPSJ), 2(1979), pp.134-142.

[TAKI80] Takizawa, M. et al.: Query Translation in Distributed Databases, Proc. IFIP, (1980), pp.451-456.

[TAKI82] Takizawa, M.: Distribution Problems in Distributed Databases - Integration and Query Decomposition, Jounal of Information Processing(JIP) (IPSJ), 5(1982), pp.139-147.

[TAKI82b]Takizawa, M. and Noguchi, S.: Non-procedural Update Interface over the CODASYL Database Systems, JIPDEC TR 11/82, (1982).

DATABASE MACHINE ACTIVITIES IN JAPAN

Syunsuke Uemura

Electrotechnical Laboratory
1-1-4, Umezono, Sakura
Ibaraki 305 JAPAN
(0298)54-5478

Research into database machines has been fairly active in Japan since 1975. There is an indication that a commercial database machine developed in Japan will appear in the near future. The ADABAS database machine and Britton Lee's IDM are already on the Japanese market.

In 1976, a national project called PIPS (Pattern Information Processing System) organized a working group to study the database machine concept. The Electrotechnical Laboratory of MITI, as the leader of the working group, implemented an experimental database machine EDC in 1978. Salient features of the EDC include the use of magnetic-bubble memory as database storage and a specialized multi-microprocessor architecture for database applications. EDC II, a successor to EDC with higher density, high performance magnetic-bubble chips (256kb/chip, 300kHz) was implemented successfully in 1980. EDC II was unveiled at the 8th IFIP World Computer Congress held in Tokyo in October, 1980 [UEMU 80].

At the same IFIP Congress, a data stream database computer was introduced by Y. Tanaka of Hokkaido University [TANA 80]. It is a network of two primitive computing modules, namely, search engine and sort engine. The search engine is a hardware implementation of a binary tree search logic. The sort engine implements heap-sort. A prototype with 4 search engines and 4 sort engines has been implemented. Research efforts to extend the system architecture for large databases are in progress [TANA 82].

GRACE is a database machine under development at Tokyo University [KITS 82]. It is a hashing based relational algebra machine especially suitable for join-intensive applications. Sort is realized by Todd's sorting hardware [Todd 78]. The design phase is being completed.

IQC (Information Query Computer) is a back-end database machine being developed at NEC Corporation [SEKI 82]. IQC can serve both as a back-end machine for a centralized database system, and as a database server for distributed database environment. IQC project is a successor to NEC's GDS (Generalized Data Subsystem) research [HAKO 77]. A Japanese computer magazine reported that NEC will announce the commercial IQC soon.

CADAM (Content-Addressable Database Access Machine) is an experimental database machine designed and being implemented by OKI Electric Industry Co. Ltd. [HIKI 81]. CADAM is a back-end hardware with conventional moving head disks, a disk cache memory and a set of special-purpose processors. Their research plans include a network of

micro-computer based database machines.

ICOT (Institute for New Generation Computer Technology) is a newly organized foundation for the research and development of the "fifth generation" computer system. (Officially, the term "fifth generation" has been replaced by "new generation".) The overall configuration of the new generation computer system includes a "knowledge base machine" supported by a relational database mechanism (machine) with VLSI technology [MOTO 82]. ICOT's research and development directions envision database machines as part of a distributed function architecture (along with high-speed numerical computing machines and so on). The long range research schedule spans for ten years.

Research activities on database machine architecture are also underway at Keio University, Hiroshima University, Yokohama National University, Yokosuka and Musashino Electrical Communication Laboratories, Hitachi Ltd. (intelligent disk, commercially available) and Toshiba Corporation (back-end mini-computer, Todd's sorting hardware).

References

[UEMU 80] Uemura, S., Yuba, T., Kokubu, A., Ooomote, R. and Sugawara, Y. "The Design and Implementation of a Magnetic-bubble Database Machine", in Information Processing 80, North-Holland (1980)

[TANA 80] Tanaka, Y., Nozaka, Y. and Masuyama, A. "Pipeline Searching and Sorting Modules as Components of a Dataflow Database Computer", in Information Processing 80, North-Holland (1980)

[TANA 82] Tanaka, Y. "A Data Stream Database Machine with Large Capacity", Proc. Int. Workshop on Database Machines, San Diego (September 1-3, 1982)

[KITS 82] Kitsuregawa, M., Tanaka, H. and Moto-oka, T. "Relational Algebra Machine GRACE", to be published.

[SEKI 82] Sekino, A., Takeuchi, K., Makino, T., Doi, T., Goto, T. and Hakozaki, K. "Design Considerations for an Information Query Computer", Proc. Int. Workshop on Database Machines, San Diego (September 1-3, 1982)

[HAKO 77] Hakozaki, K., Makino, T., Mizuma, M., Umemura, M. and Hiyoshi, S. "A Conceptual Design of a Generalized Database Subsystem", Proc. 3rd Int. Conf. on VLDB (1977)

[HIKI 81] Hikita, S., Yamazaki, H., Hasegawa, K. and Matsushita, Y. "Optimization of the File Access Method in Content-Addressable Database Access Machine (CADAM)", Proc. AFIPS 1981 NCC, AFIPS Press (1981)

[MOTO 82] Moto-oka, T. (ed.) "Fifth Generation Computer System", Elsevier Science Publishing (1982)

[TODD 78] Todd, S. "Algorithm and Hardware for a Merge Sort Using Multiple Processors", IBM J. Research and Development, 22, 5 (1978)