TM-0008

A Relational Database Machine "Delta"

(Translated form IPSJ)

by
Takeo Kakuta, Nobuyoshi Miyazaki,
Shigeki Shibayama, Haruo Yokota
and Kunio Murakami

May, 1983

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Relational Database Machine "Delta" (I)

---Background and Objectives of its Development---

Takeo Kakuta, Nobuyoshi Miyazaki, Shigeki Shibayama,
Haruo Yokota, Kunio Murakami

Institute for New Generation Computer Technology (ICOT)

1. Introduction

The Fifth Generation Computer System (FGCS) project's major aim is to carry out researches and develop a prototype of a knowledge-information-processing oriented computer based on new technology to meet the anticipated requirements of the 1990's, such as an inference function and an intelligent and interactive function using a knowledge base. The FGCS project is developing a Relational Database Machine (RDBM) called Delta in the first research and development stage of Knowledge Base Machines. This paper will describe the background and objectives of the development and outline of the functions of Delta.

2. Background of the Database Machine Development

As the database is extended in both its application field and size, the requirements for a database system with a higher transaction processing capability arises. To implement such a system, it is necessary to develop a database machine as a dedicated processor for the system. Various experimental machines [1],[2] and several commercial database machines have been introduced.

In the knowledge information processing field, high speed knowledge base searching and knowledge operation functions are required because, in addition to mere storage of a large amount of knowledge, inference is performed using the knowledge. Developing a Relational Database Machines as a part of the FGCS project will prove effective for not only establishing the basic technologies for the research and development of a knowledge information processing but also implementing large-volume database machines in the future.

3. Development Step of a Knowledge Base Machine

(1) Initial Stage: The knowledge base system functions are considered to be divided into inference functions and database functions. For each class of function, we plan to develop software and hardware research and development support system by integrating two experimental subsystems - a sequential inference machine and a relational database machine.

(2) Middle and Final Stages: By gradually amalgamating the inference functions with the knowledge base functions, a highly parallel knowledge base machine will be investigated and

developed in the middle stage, and a knowledge information processing system, in the final stage.

4. Goals of the Relational Database Machine Development

The development of Delta has the following three items as the major targets:

(1) To provide a computer system with multiple Sequential Inference Machines (SIM's) and Delta connected via a local area network (LAN) as a prototype tool for software and hardware research and development support by the end of the initial stage.

(2) To perform various experiments and measurements, to collect evaluation data for the research and development of Knowledge Base Machines, to extract from the results necessary functions for supporting knowledge operations based on relational algebra operations and to make suggestions on expansion and improvements of architecture required for implementing these functions.

(3)To check and evaluate processing algorithms of RDBMs and make a research into highly parallel database machines.

5. Delta Overview

5.1 Functional Design Requirements

(1) Delta must be capable of storing databases with a total capacity of about 10 Giga bytes, as well as performing high-speed and efficient search, update and other relational algebra operations.

(2) Delta is connected to multiple Sequential Inference Machines (SIM) via a local area network, or otherwise directly to one. With the hardware and software (including firmware) within Delta and the relational database management software working in a SIM, Delta must be able to efficiently support queries sent to external database from a user program written in the kernel language, the logic programming language of ICOT.

Fig.1 shows the relationship between Delta and SIMs.

(3) In order to satisfy the needs of various experiments conducted for the research and development of knowledge base machines and a high speed parallel knowledge operation mechanism, Delta must be organized to provide data collection functions as well as the architectural flexibility for Hierarchical Memory expansion and additional RDB Engine attachment.

5.2 Characteristics of Delta's Architecture

Fig.2 shows a conceptual diagram of the interfaces between Delta and the SIM software illustrated from the architectural standpoint. Delta consists of five subsystems as shown in Fig.3.

(1) Queries from a user program written in the kernel language or a natural language are converted into commands at the relational

algebra level. In other words, when a query is sent to an external database from a user program written in the kernel or a natural language, the relational database management software in SIM converts it to a relational algebra level command and accesses Delta via a LAN.

(2) The load of the software in the RDBM's Control Processor is distributed by delegating data manipulation functions and memory management functions, which are equivalent to the operating system for an ordinary database management system, to the dedicated hardware.

(3) Relational Database Engine (RDBE) is a dedicated processor which performs a hardware-oriented relational algebra processing algorithm based on sort and merge operation. Under control of Control processor, it efficiently performs relational algebra command processing on relation data supplied in stream form from Hierarchical Memory.

(4) Hierarchical Memory (HM) stores a large amount of relations and directory/dictionary information. HM consists of the following three layers with different capacity and access time; from bottom to top.

(a) moving head disk
(b) silicon disk (SDK)
(c) high speed working buffer

(5) Control Processor (CP) performs resource management, directory/dictionary management, concurrent control and other processes for RDBE and HM. The control software for entire Delta resides in CP.

(6) Interface Processor (IP) is responsible for the LAN adapter interface functions with CP and HM. It passes to CP a relational database query command sent from SIM via a LAN, and sends back the resulting relation to the request user.

(7) Maintenance Processor (MP) has status monitoring, initialization, diagnostics and data collection functions for Delta.

6. Conclusion

RDBM Delta is currently in the stage of functional design. We will report the results occasionally, when its design is proceeded to the detail stage or when evaluation and experiments are performed after the completion of Delta system.

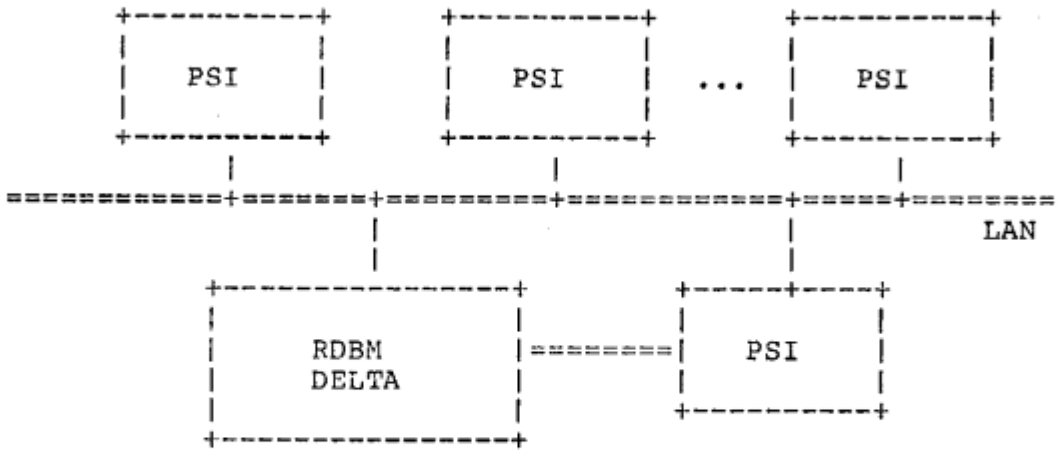

Fig.1 Relationship between Delta and SIM's (PSI's)

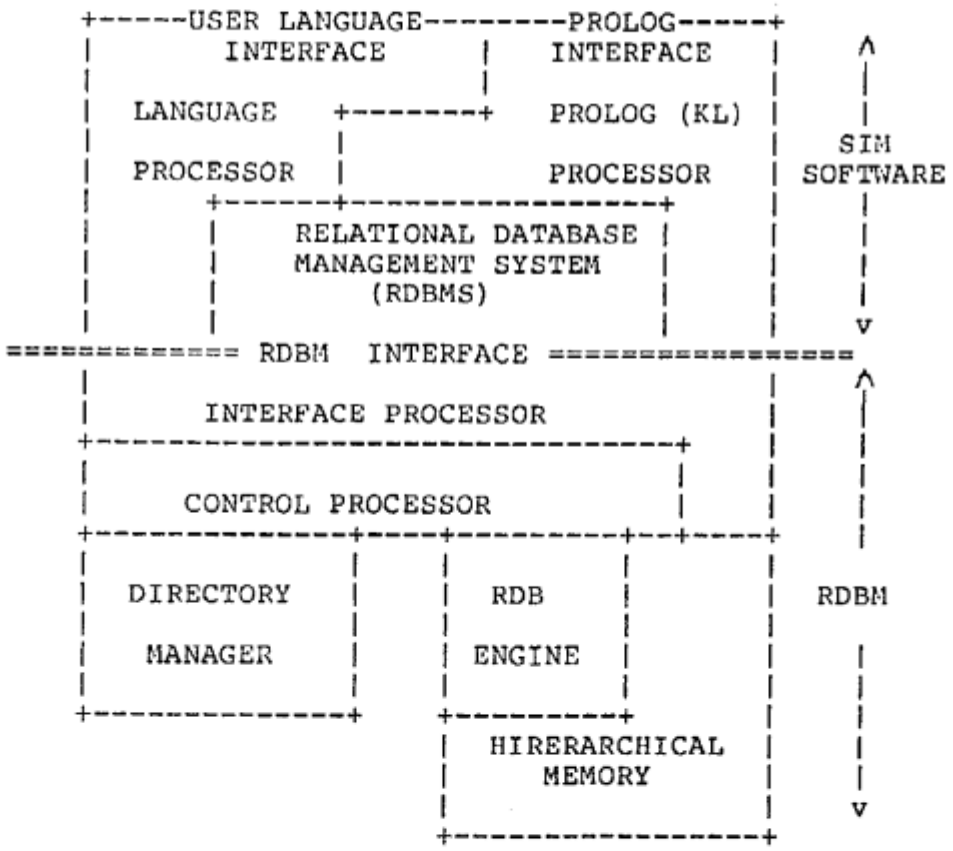

Fig.2 Conceptual Diagram of the Interface Between Delta and the SIM Software Illustrated from the Architectural Standpoint

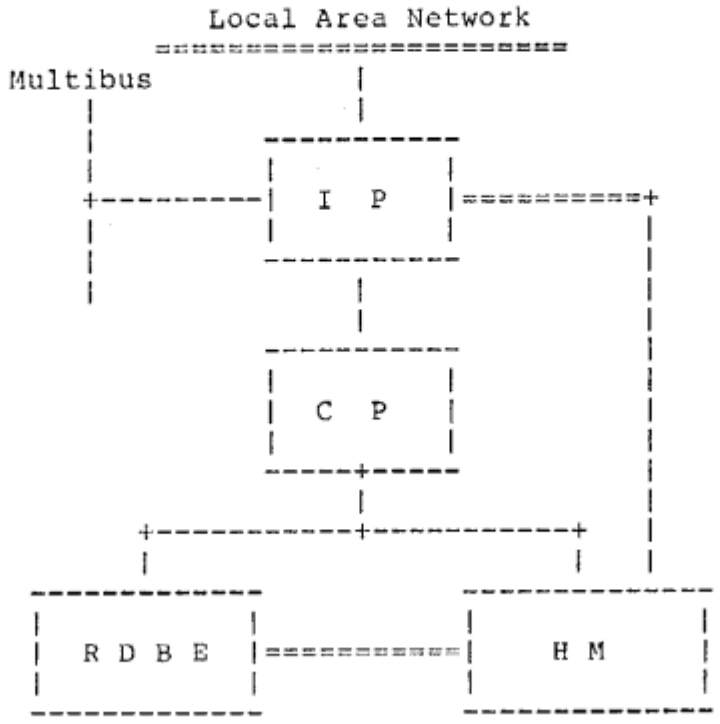

Fig.3 Functional Diagram of Delta

7. References

[1] H.Schweppe et al. ," RDBM-A Dedicated Multiprocessor Systems for Data Base Management " , Proceeding of the International Workshop on Database Machines,August,1982

[2] F.Bancilhon et al. ," VERSO: A Relational Back-End Data Base Machine " , Proceeding of the International Workshop on Database Machines,August,1982

A RELATIONAL DATABASE MACHINE DELTA (II)
-BASIC ARCHITECTURE-

Shigeki Shibayama, Takeo Kakuta, Nobuyoshi Miyazaki,
Haruo Yokota, Kunio Murakami

Institute for New Generation Computer Technology (ICOT)

1. Introduction

The Relational Database Machine "Delta" planned for development by the end of the initial three-year-long stage of the Fifth Generation Computer System (FGCS) project is characterized as follows:

1) Stores the database (knowledge base) for an application system and works as a development tool for the system.
2) Functions as an experimental machine to conduct research and development of knowledge base machines.

Functions emphasized in the development stage include:

1) Communicates with the personal inference machine (PSI) via a local area network (LAN) and receives a query to a database.
2) Provides a database machine with a memory capacity in the gigabyte order and a hardware processing mechanism.

These objectives must be accomplished in conjunction with the development of the relational database management system (RDBMS), the software of the PSI.

2. External Interface

A relational-algebra-based command set has been selected as the interface for Delta for several reasons:

1) If a body in a prolog program (clause) is found to exist in the external rather than internal database when the clause is executed, it is possible to postpone evaluation of the body and, when a number of such bodies are accumulated, to issue a query for the entire batch of bodies. In this case the use of relational algebra makes it easier to express the query [1],[2].
2) When performing a relational model operation with hardware in the database machine, relational algebra well matches the algorithm we use.
3) Relational algebra can provide a sufficiently high-level non-procedural interface.

The command set so far established is included in [4]. Since Delta is shared among multiple users, it must provide external interface functions such as concurrent execution and control of transactions, data recovery and backout.

3. Basic Architecture of Delta

Fig.1 shows the basic configuration of Delta. The function of each subsystem shown in the figure is described below.

1) Control Processor (CP)

Primarily the Control Processor manages process execution controlling all of Delta. It analyzes a command-tree given from hosts (SIM's) and expands it into subcommands for the hierarchical memory (HM) and RDB engine. While analyzing the command-tree, the CP references a dictionary to obtain necessary information for concerned relations. It also performs concurrent execution control and backout. The CP software consists of two layers as follows:

i) DB Management level
ii) Unit-Command Process level

The Database Management level is similar to the data management software offered by a conventional DBMS, and performs major part of concurrent execution control, command analysis, command-tree processing and others. In contrast the Unit-Command Process level executes the command set of Delta and is mainly responsible for management and control of hardware resources. To facilitate development, we have decided to compose the CP from an existing minicomputer and Operating System performing inter-process communications.

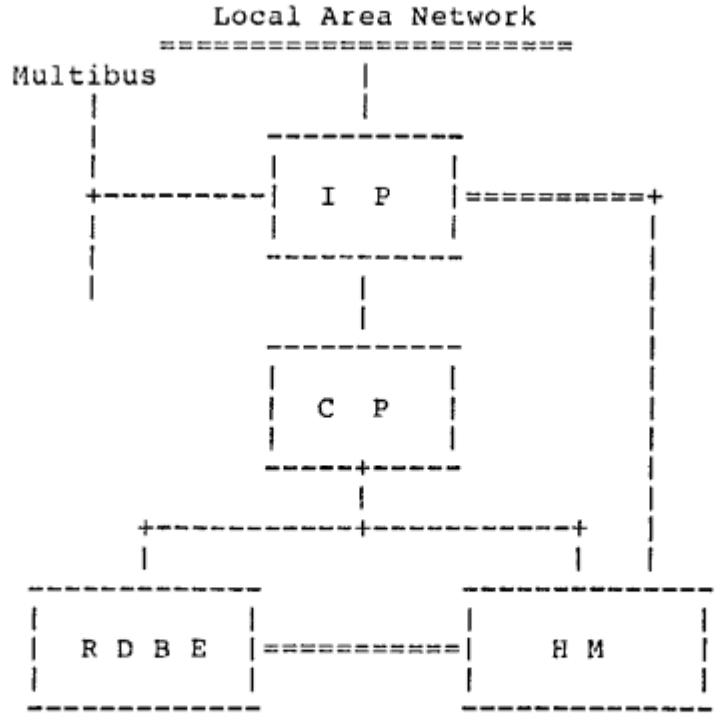

Fig.1 Basic Delta Block Diagram

2) RDB engine (RDBE)

The RDBE is a piece of hardware which executes search-related Delta commands consisting mostly of relational algebra commands. Th RDBE has a low-level subcommand set associated with Delta's command set. This subcommand set reflects the RDBE hardware algorithm. The RDBE's processing algorithm uses a pipeline merge sorter to sort an input attribute on-the-fly with the transfer of the attribute, and then passes the sorted attribute to the next stage, the relational operation unit (ROU), to execute a subcommand. We plan to provide Delta with multiple units which execute this algorithm and to interconnect them with a network so that we can experiment the parallel processing of a command-tree.

3) Hierarchical Memory (HM)

The Hierarchical Memory is the actual database store. Logically, the HM consists of the following layers hierarchically structured:

i) Permanent Relation Layer (PRL)
ii) Non-Permanent Relation Layer (NPRL)

The PRL corresponds to permanent relations whereas the NPRL corresponds to relations which are temporarily generated and cleared in a transaction. Physically, the HM is divided into the following hierarchically-structured layers:

a) Moving-Head Disk Layer (MHDL)

b) Silicon Disk Layer (SDL)

c) Working Buffer Layer (WBL)

Fig. 2 shows the relationship between the logical and physical configurations. Both the PRL and NPRL are mapped so that they have their own virtual space in the physical layer. The WBL is not shown in the figure.

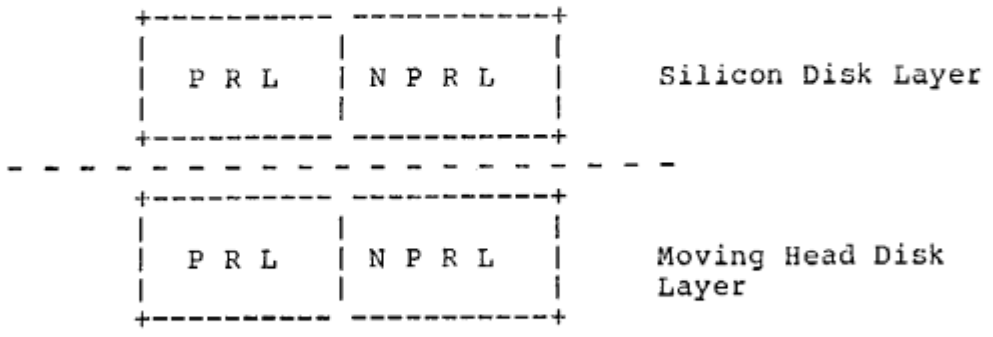

Fig.2 Relationship between the Logical and Physical Configurations

Another important function of the HM is to provide a working space for tuple reconstruction. As described in the next section, Delta stores, as the internal schema, relations by attribute (column). Therefore, when outputting a relation in the form of tuples, the join operation by means of a tuple identifier (tid) is required. The algorithm for the join operation requires multiple buckets according to the length of attributes reconstructed on the Silicon Disk, vertically stacks attributes in the order in which they were sorted by their tid's, and reads the buckets horizontally.

4) Interface Processor (IP)

The Interface Processor connects Delta to LAN environment and to the Multibus. It can directly access the HM as well as receive input and send resulting relations independently of CP. The Multibus Interface (MBI) offers a bus via which accesses to Delta are accomplished with less overhead than the LAN.

4. Internal Schema

Since Delta uses the RDBE to scan part of relations, it does not require an index at the tuple level such as an inverted file. To avoid scanning all relations every time, Delta has a directory consisting of a group of attribute clusters. The first stage of the directory is divided based on the range of attribute values and the next stage is divided based on the range of tid values. Fig.3 shows the configuration of the internal schema. Since both are clustered by value, reconstruction can be done by the sorter in the RDBE.

5. conclusion

This paper described the basic architecture of Delta. Further evaluation of the approaches taken in Delta will be reported through simulations and so on.

6. References

[1] Kunifuji, Yokota, et al., "Interface between Logic Programming Language and Relational Database Management System (1) -- Basic Concepts --", 26th National conference, Information Processing Society of Japan, 5C-9, 1983.

[2] Yokota, Kunifuji, et al., "Interface between Logic Programming Language and Relational Database Management System (2) -- Implementation --", 26th National conference, Information Processing Society of Japan, 5C-10, 1983.

[3] Kakuta, Miyazaki, Shibayama, Yokota, Murakami, "A Relational Database Machine "Delta" (I)", 26th National conference, Information Processing Society of Japan, 4F-6, 1983.

[4] Yokota, Kakuta, Miyazaki, Shibayama, Murakami, "A Relational Database Machine "Delta" (III)", 26th National conference, Information Processing Society of Japan, 4F-8, 1983.

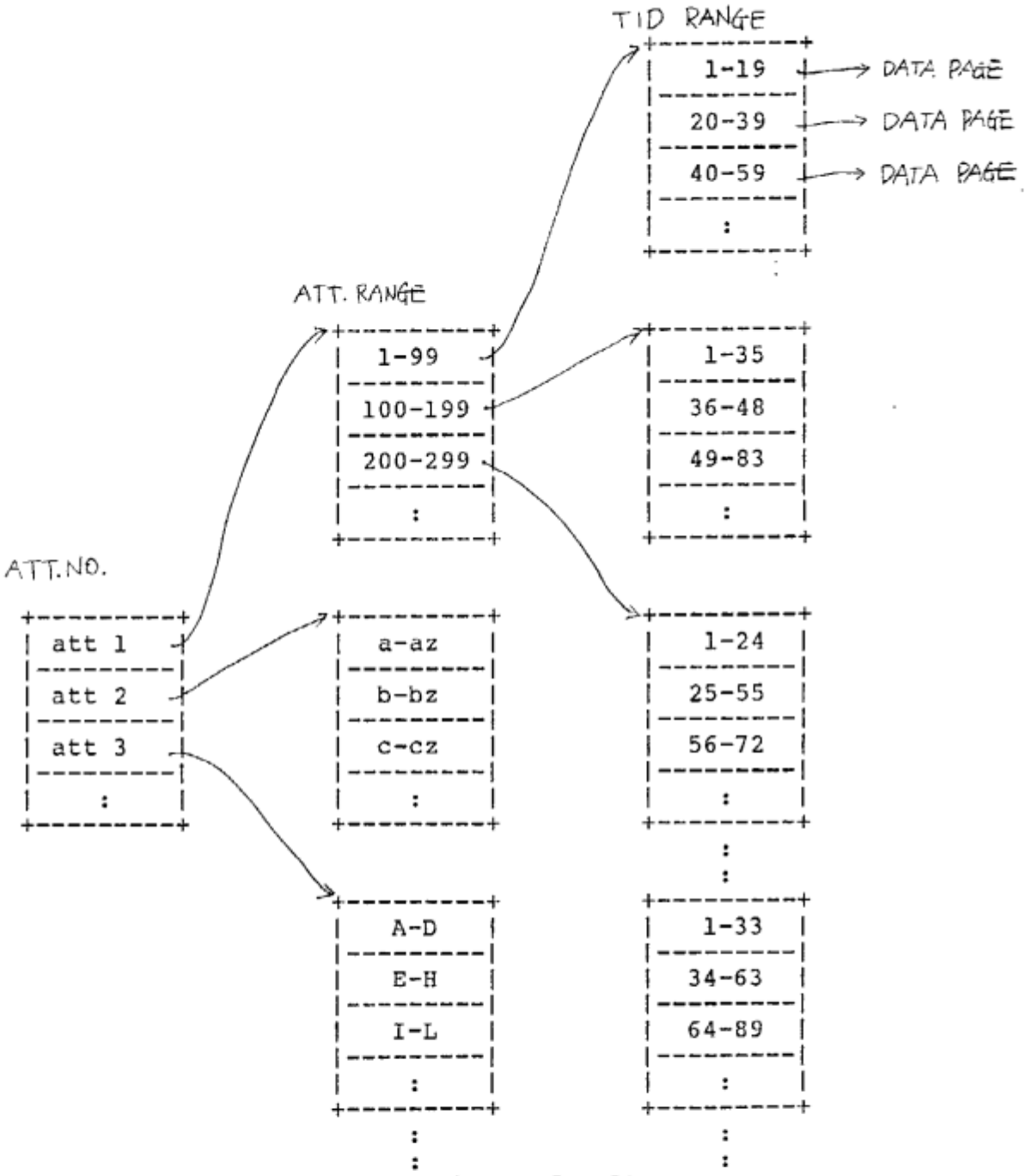

Fig.3 Internal Schema of Delta

A Relational Database Machine "Delta" (III)

--- Command Set ---

Haruo Yokota, Takeo Kakuta, Nobuyoshi Miyazaki,
Shigeki Shibayama, Kunio Murakami

Institute for New Generation Computer Technology
(ICOT)

1. Introduction

Relational Database Machine "Delta" currently under development will be connected to host machines and function according to a request from the host machines. We have established a relational-algebra-based command set as a logical interface between Delta and the host machines. This paper describes a Prolog-based simulation which has been carried out to validate the command set.

2. Basic Concept

Since the main purpose of the simulation is to check the appropriateness of the command set, we mostly check the command operations but do not intend to evaluate the machine's performance. However, data collection to find the frequency of command execution on this simulator can be performed on a limited scale, since it is easily realized with the addition of a few programs.

It is difficult to judge whether the command set really functions in actual operation by simply simulating command execution at the relational algebra level. Therefore, we have decided to make queries using a database query language which is a higher-level one than the relational algebra, to convert the queries into a Delta command sequence, and then to simulate the command sequence using a Delta simulator. As a query language, we have selected SEQUEL2 [1] currently widely used as a relational database query language. Fig. 1 shows the configuration of the entire simulation process.

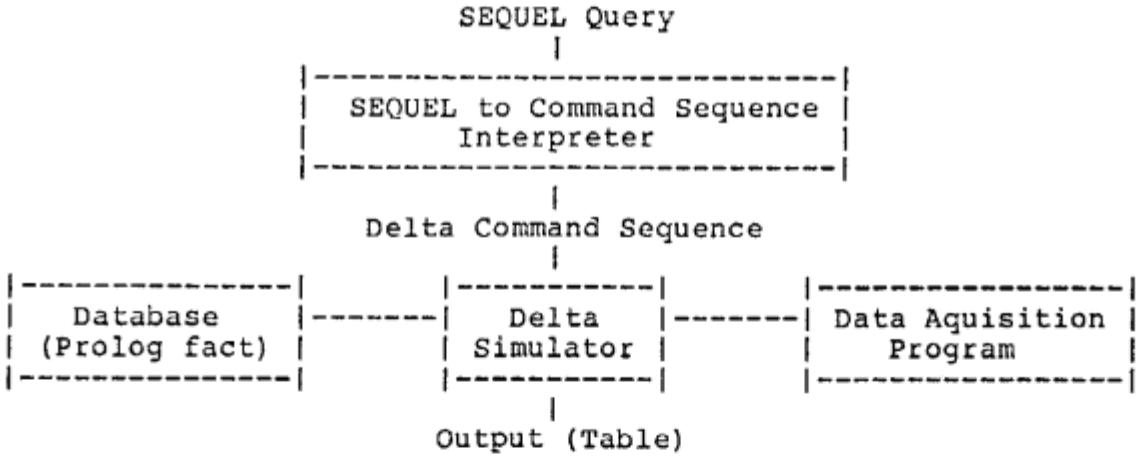

Fig. 1 Configuration of the Simulation Process

We have decided to write the simulation system in Prolog because of several reasons as follows:

(a) Due to its high affinity to relational database, Prolog is suitable as the simulator for a relational database machine.
(b) With the pattern matching facility of Prolog, the interpreter can be written in a fairly easy manner.
(c) Since Prolog has been used as the base of the Kernel language for the system to be developed in the FGCS project, an interpreter once written in Prolog can be used, with little modification, as a basis for a future database query system.

Note that the Delta simulator will be used when we investigate the interface between a logic programming language and a relational database management system [2].

3. Delta Simulator

A Prolog program consists of a set of rules and a set of facts. The set of facts can be considered a small-scale relational database. In other words, it is possible to correspond the predicate name of a fact with the name of a relation, and each argument with an attribute of the relation. In this simulator we have decided to use Prolog facts directly as a database because of two reasons as follows:

i) Our simulator which does not evaluate the machine's performance, etc. can ignore the detailed database manipulation inside a real machine, such as how database is actually stored (internal schema).

ii) It is sufficient for our objective to see how each command is executed on a small-scale database of Prolog facts.

Thus, our simulator can be very easily implemented by directly using facts as a database. Since it is necessary to distinguish facts as a test database from those of a program, however, we have decided to prepare a relation called "relations" which has the name of a relation and the number of attributes as arguments. Although each attributes of a relational database has its name, a Prolog fact has nothing but the positional relationship of each argument. Therefore, we have decided to prepare a relation called "attributes" which manages the name of a relation, the name of each attribute in the relation, the position of the argument corresponding to each attribute, and the data type and length of each attribute.

For example, from a relation name "dept" with four attributes (deptno, dname, loc, empcnt), the following set of facts are produced:

```
relations(dept,4).
attributes(dept,deptno,1,integer,7).
attributes(dept,dname,2,char,16).
attributes(dept,loc,3,char,16).
attributes(dept,empcnt,4,integer,8).
dept(10,research,tokyo,0).
dept(20,sale,osaka,0).
        :
        :
```

Commands accepted by our simulator are the same as those for Delta, except that they agree with Prolog's goals in the format. For example, the following command is used for projection:

```
projection(dept,[1,3],newtable).
```

where the second argument [1,3] shows the first attribute (deptno) and the third attribute (loc) of the relation named "dept." Commands like this example must process all the tuples of a relation. This process is equivalent to one performed in Prolog on all facts with the same predicate name, and can easily be implemented using Prolog's backtrack facility. The projection example above can be realized with, for example, the following

program:

```
projection(SRN,ATTN,RRN)   :-
      get_predicate(SRN,SRP),
      select_attributes(ATTN,SRP,RRAG),
      insert_tuple(RRN,RRAG),
      fail.
```

Table 1 lists commands implemented for our simulation purpose.

Relational Algebra Commands	Aggregate Function Commands
Projection	Count
Restriction	Summation
Selection	Maximum
Natural-Join	Minimum
theta-Join	Average
Set Operation Commands	**Check Commands**
Union	Equal
Difference	Contain
Intersection	**Input/Output Commands**
Cartesian-Product	Get
Update Commands	Put
Delete	**Others**
Update	Sort
Insert	Unique
Relation Definition Commands	Group-by
Create-Relation	Select-Group
Purge-Relation	Copy
Rename-Relation	

Table 1 A list of Delta Commands

4. SEQUEL Interpreter

SEQUEL is equipped with unified functions consisting of query, data manipulation, data definition and data control. Our interpreter simulates most of its functions except some secondary functions such as VIEW management, consistency control and security control. To follow the original syntax [1] as much as possible, syntax modification to facilitate simulator programming in Prolog is minimized. In syntax analysis, the BNF expression can be maintained almost exactly in the original form in the interpreter, by effectively using Prolog's unification function. In addition to this, since it is easy to correspond SEQUEL statements to Delta commands, we have used list processing to perform code generation (i.e. command sequence generation in our case) simultaneously with syntax analysis.

For example,

```
select  deptno,loc.
from    dept.
where   empcnt=[0].
```

for these SEQUEL queries, the interpreter generates Delta commands as follows:

```
restriction(dept,[4]=0,temp1).
projection(temp1,[1,3],temp2).
get(temp2).
```

5. Conclusion

This paper described the Prolog-based simulator for Delta command execution and the interpreter which converts a SEQUEL query into a Delta command sequence. Since Delta uses an approach of storing relations by attribute clusters, we will prepare a simulator which makes use of such an internal schema and examine it. Also we will add VIEW management, consistency control and other secondary functions to the SEQUEL interpreter.

References:

[1] D.D.Chamberlin, et al., "SEQUEL 2:A Unified Approach to Data Definition, Manipulation, and Control", IBM J. RES. DEVELOP., November, 1976.

[2] Yokota, Kunifuji, et al., "Interface between Logic Programming Language and Relational Database Management System (2) -- Implementation --", 26th National conference, Information Processing Society of Japan, 5C-10, 1983.