IMPLEMENTATION OF A TIME EXPERT

IN A DATA BASE SYSTEM

by

Ricky Overmyer and Michael Stonebraker
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF CALIFORNIA
BERKELEY, CA.

ABSTRACT

This paper reports on the design and implementation of a time expert for a relational data base system. It demonstrates that a sophisticated expert can be written easily and cause minimal performance degradation. As a result, extending a data base system to support user defined data types is an easy operation.

I INTRODUCTION

A feature often requested in data base management systems is user defined data types. Although most data base systems provide support for integers, real numbers and character strings, they typically do not allow data types such as money, length, bit-string, etc. User who have data fields with additional semantic meaning must support such types in application code. Clearly, allowing new data types in a data manager would be a useful construct.

In a previous paper [STON80] we indicated the structure of a data base expert which can be used for this purpose. Although not restricted to the definition of new types, this is one important use of experts. New types can also be obtained by the inclusion of abstract data types in a data manager; see [BALD81] for an approach along these lines.

In order to test the utility of experts we designed and built a sophisticated time expert. This paper reports on our experience. First, in Section II we review the definition of an expert. Then in Section III we discuss the capabilities of the time expert we constructed. Lastly, we close with some benchmark timings and the magnitude of the project in man-weeks and lines of code.

II EXPERTS

The function of an expert is embodied in four procedure calls which are invoked at various places during the

processing of a data base command. Since the expert we describe is embedded in INGRES [STON76], we will use that environment for our presentation. However, the mechanism presented could be put in any data manager presumably with similar results.

An expert field is indicated during creation of a relation, for example:

```
CREATE EVENT (ename = c20, time = time-expert)
```

Here, EVENT contains two fields, a twenty byte character string and a field controlled by the time expert. When an expert is registered with INGRES, the actual length of the stored representation for the time field must be indicated.

A user can now run ordinary QUEL commands against the EVENT relation. For example:

```
APPEND TO EVENT ( ename = "lunch",
                  time  = "12:00-1:00")
```

Before INGRES inserts "12:00-1:00" into the data base, it calls the time expert as follows:

```
code-value = time-expert-1 ("12:00-1:00")
```

The return code is a fixed length object which is stored by INGRES in the data base. Later a user may query the EVENT relation as follows:

```
RANGE OF E IS EVENT
RETRIEVE (E.time) WHERE E.ename = "lunch"
```

This command can be processed almost to completion by the normal INGRES code. However, a code for the time of "lunch" cannot be returned to the user. Consequently, INGRES must call the time expert to find the external representation as follows:

```
string-return = time-expert-2 (code-value)
```

These two procedure calls support "external" to "internal" transformations for data values in a manner similar to that used in MacAIMS [GOLD70].

Next, suppose a second tuple is added to EVENT as follows:

```
APPEND TO EVENT ( ename = "fire drill",
                  time  = "12:50")
```

and then suppose the data base is queried to find out if the fire drill happens during lunch, i.e.

```
RANGE OF F IS EVENT
RETRIEVE (F.ename) WHERE F.time = E.time
                           and
                       E.ename = "lunch"
```

This query will be processed to the point where INGRES has two codes, one for the time of lunch and one for the time of the fire drill. It must then ask the time expert if these codes are equal, i.e.

```
boolean-return = time-expert-3 (code-1, code-2, = )
```

Basically, time-expert-3 is responsible for the definition of the QUEL operator "=" as well as the other comparison operators {<, <=, >, >=, !=}.

The last procedure call concerns arithmetic. For example, we could move the fire drill ahead one hour by

```
REPLACE E(time = E.time + "1 hour") WHERE
                  F.ename = "fire drill"
```

INGRES can process this command to completion except for the necessity of doing the arithmetic sum of a code value for "fire drill" and a code value for "1 hour." Hence, it calls the expert:

```
code-return = time-expert-4 (code-1, code-2, + )
```

This procedure call is responsible for the definition of "+," as well as all the other legal arithmetic operators in QUEL.

In summary, an expert is a collection of four procedure calls that have been registered with INGRES. At run time, each routine is called when appropriate. Currently expert routines are linked into INGRES as subroutines. If protection was an issue they could be run in a separate process and the interprocess message facility used to provide protected communication. However, "non trusted" experts incur the performance penalty of replacing each subroutine call with an interprocess message.

III DEFINITION OF THE TIME EXPERT

In this section we discuss the external formats supported by our time expert. They are similar in functional sophistication to those proposed in [KAHN75]. Hence, we can obtain a rather elegant expert quickly and simply.

3.1 Simple Dates

The time expert accepts a simple date of the form:

(day-of-week month day-of-month hour minute second year)

For example,

Sunday June 2 8:15:07 1978

is a legal time. Any of the components may be omitted and the months and day of week can be abbreviated. Lastly, both a 12 hour clock and a 24 hour clock are accepted.

3.2 Range of Dates

Our expert accepts any range of values whose endpoints are simple dates. For example,

Sunday June 2 - Sunday June 9 17:24

is a legal external time.

3.3 Relative Ranges of Time

In addition to ranges of times which are absolute, the expert allows ranges of time which are relative to the current time of day or day of week. For example, "today," "last week," "7 years ago," and "9 weeks from now," are all allowed.

3.4 Intervals of Times

For example, "7 days", "4 years" and "99 minutes" are acceptable external times.

3.5 Repeating Intervals

A repeating interval, such as "Sunday", "January", or "12:00 - 1:00", is also allowable. Both "lunch" and "fire drill" in the preceding section were times in this category.

A complete BNF of the initial grammar accepted by the time expert is given in Appendix 1. Moreover, since the legal times are stored in a relation in the data base it is possible for an (intelligent and persistent) user to directly modify this relation using QUEL to extend (or contract) the vocabulary of the time expert.

IV IMPLEMENTATION

Our time expert stores time internally in INGRES as a 32 byte character string. The first 16 bytes are the lower bound of a time range, while the latter 16 bytes are the upper bound. A simple date has a blank value for the upper bound. The coding is:

year:	4 bytes
month:	2 bytes
day:	2 bytes
day-of-week:	1 byte
am/pm:	1 byte
hour:	2 bytes
minute:	2 bytes
second:	2 bytes

It is clear that this representation could be substantially compressed; however, this coding was an especially easy one for which to write arithmetic routines.

The first two calls to the time expert simply do the appropriate conversion from external representation to internal representation and back; i.e. between string format and 32 byte format.

The third procedure call correctly handles all comparison operators. Simple dates and ranges of time can be compared. The answer to any such comparison is "yes", "no" or "maybe". The latter answer can only occur when one range overlaps another. The time expert reports "yes" when ranges of time overlap. Implementing any notion of fuzziness was not attempted.

Arithmetic operators are supported by the fourth procedure call with one exception. It is impossible to add or subtract one range from another. Functions like

"5 weeks" - "1 week"

are currently unsupported. This is a shortcoming of the existing implementation.

The TIME_EXPERT relation assists the time expert in doing the conversion between external and internal representations. It has the following structure:

TIME_EXPERT(EXTNAM1, INTCODE, EXTNAM2, OP)

This relation stores the internal code, INTCODE, for each external name, EXTNAM1. The coding of values for INTCODE allows the substitution of the current clock time into a code which is produced as a result of the first procedure call. For example, the following QUEL command might be executed.

```
APPEND TO EVENT( ename = "my deadline",
                 time = "today")
```

The time expert produces a code for "today" by reading the system clock and then truncating fields from the resulting simple date.

In addition, one of the external names is a code for any particular day of the week. Hence, a single tuple in TIME_EXPERT allows one to code all seven days of the week. The same technique is user for years,, months of the year, etc. Lastly, EXTNAM2 and OP are designed to allow times to be arithmetic combinations of other times. For example, "yesterday" is defined to be:

"today" - internal code for 1 day

Here, OP stores the "-" operator while EXTNAM2 contains the external name "today". As a result, any user who understands the coding conventions for these fields, can add new entries and enlarge the vocabulary of the time expert.

One drawback of the current INGRES implementation is that one cannot run a QUEL command from within INGRES. Consequently, our time expert must read the TIME_EXPERT relation by doing access method calls.

IV CODING COMPLEXITY AND BENCHMARKING RESULTS

4.1 Implementation Effort

The implementation effort consisted of writing the four procedures and modifying INGRES to make the appropriate calls. The hooks to INGRES occurred in a total of five places and involved adding or changing a total of 62 lines of code. The time to make these modifications was inconsequential. However, 3 man weeks of familiarization with the INGRES code was required before the modifications could be attempted.

The time expert was then written. It encompasses 843 lines of code and was written in 6 man weeks. In particular, the four routines had the following sizes:

encode	376
decode	103
comparison	134
arithmetic	140

The remainder of the code was global variables. Because the time expert is isolated from the DBMS routines, debugging was a relatively easy task.

4.2 Benchmark

In order to assess the performance degradation due to the inclusion of an expert, we constructed the benchmark indicated in Figure 1 and ran it against an ordinary INGRES relation with time as a character string field. Then, we ran the same commands against INGRES with a time expert. Figure 1 also indicates the percentage degradation resulting from the time expert. The two relations EVENT10 and EVENT180 have respectively 10 and 180 tuples. The EasyAppends add either 1 or 10 tuples to EVENT10. Moreover, the coding for "today" requires a single call to the TIME_EXPERT relation. On the other hand, in HardAppends both "yesterday" and "tomorrow" are defined in the TIME_EXPERT relation as offsets from "today". Hence two accesses to the expert's data base are required to code each of these fields. Lastly, the two Retrieves simply decode the time field in each tuple of EVENT10 and EVENT180 respectively.

It can be seen that the degradation in performance varies between 5 and 38 percent. As expected, the maximum degradation occurs when INGRES is processing HardAppends which require multiple reads to the TIME_EXPERT relation to code the external string. In other cases however, the performance degradation is more modest.

V CONCLUSIONS

It can be seen that a complex expert is straightforward to write and runs with reasonable performance. It is expected that a multitude of experts could be constructed with similar results.

COMMAND	TUPLES	QUEL COMMAND	OVERHEAD
EasyAppend	1	append to EVENT10(time="today")	23.8
EasyAppend	10	range of E is EVENT10 append to EVENT10(ename = E.ename time = "today")	14.7
HardAppend	1	append to EVENT10(time = "yesterday-tomorrow")	38.1
HardAppend	10	range of t is EVENT10 append to EVENT10(ename = E.ename time = "yesterday-tomorrow")	26.5
Retrieve	10	range of E is EVENT10 retrieve(E.time)	4.8
Retrieve	180	range of E is EVENT180 retrieve(E.time)	30.2

Figure 1: Benchmark Results

REFERENCES

[BALD81] Baldwin Christopher, "Extending a Data Base System With Abstract Data Types," Master of Science Thesis, University of California, Berkeley, Ca., August 1981.

[GOLD70] Goldstein, R. and Strnad, A., "The MacAIMS Data management System," Proc. 1970 ACM-SIGFIDET Workshop on Data Description and Access, Houston, Texas, November 1970.

[KAHN75] Kahn, K. M. "Mechanization of Temporal Knowledge," Project MAC, MIT, September, 1975.

[STON76] Stonebraker, M. et. al., "The Design and Implementation of Ingres," ACM Transactions on Database Systems, Vol 1, No 3, September, 1976.

[STON80] Stonebraker, M., and Keller, K., "Embedding Expert Knowledge and Hypothetical Data Bases Into a Data Base System," Proceedings of 1980 ACM-SIGMOD Conference on Management of Data, Santa Monica, Ca., May 1980.

Appendix 1

BNF Description of Initial Vocabulary

```
TIME     ::= TIME1 || TIME2
TIME1    ::= PART1 PART2
PART1    ::= TIMCOMB || TIMEXP
PART2    ::= - PART1 || ""
TIMCOMB  ::= DOW MTH DAT CLK YR
DOW      ::= Sun||...||Sat || Sunday||...||Saturday || ""
MTH      ::= Jan||...||Dec || January||...||December || ""
DAT      ::= 1||...||31 || ""
YR       ::= 0||...||9999 || ""
CLK      ::= AP || HR1 AP || HR1:MIN AP || HR1:MIN:SEC AP ||
             HR2 || HR2:MIN || HR2:MIN:SEC || ""
HR2      ::= 0||...||24
HR1      ::= 1||...||12
MIN      ::= 00||...||59
SEC      ::= 00||...||59
AP       ::= AM || PM
TIMEXP   ::= YRS || MTHS || WKS || DAYS || HRS || MINS ||
             N2MTHS/NYR || N2MTHS/NDAT/NYR
YRS      ::= this year || next year || last year ||
             NYR years ago || NYR years from now
NYR      ::= 0||...||9999
MTHS     ::= this month || next month || last month ||
             NMTH months ago || NMTH months from now
NMTH     ::= 0||...||99
N2MTHS   ::= 1||...||12
WKS      ::= this week || next week || last week ||
             NWK weeks ago || NWK weeks from now
NWK      ::= 0||...||9
DAYS     ::= today || yesterday || tomorrow ||
             NDAY days ago || NDAY days from now
NDAY     ::= 0||...||99
NDAT     ::= 1||...||31
HRS      ::= this hour || last hour || next hour ||
             NHR hours ago || NHR hours from now
NHR      ::= 0||...||99
MINS     ::= this minute || last minute || next minute ||
             NMIN minutes ago || NMIN minutes from now
NMIN     ::= 0||...||99
TIME2    ::= NYR years || NMTH months || NWK weeks ||
             NDAY days || NHR hours || NMIN minutes
```