Mnesia - An Industrial DBMS with Transactions, Distribution and a Logical Query Language

Hans Nilsson and Claes Wikström
{hans,klacke}@erix.ericsson.se

Computer Science Laboratory
Ericsson Telecom AB
S-125 26 Stockholm, Sweden

Abstract

Mnesia is a full DBMS made for telecommunications industrial needs. It has distributed transactions, fast real time lookups, crash recovery and a logical query language. The DBMS is written in the functional language Erlang *which is also the intended applications language. It already has real users developing products.*

1 Introduction

Mnesia is a multiuser Distributed DBMS (Database Management System) specially made for industrial telecommunications applications written in the symbolic language Erlang[AVWW95].

In telecommunications applications there are needs different from the features provided by traditional DBMS's. The applications we now see implemented in the Erlang language need a mixture of a broad range of features. The most extreme among those are:

1. fast soft real-time key-value lookup
2. complicated non-realtime queries mainly for operation and maintenance
3. distributed data due to distributed applications
4. high fault-tolerance
5. dynamic re-configuration
6. variable-length records

In the current first version of Mnesia we primarily use well-known algorithms. In following versions we will begin to experiment with alternative solutions.

Many telecommunications applications are structured in such way that there are two different sets of data. One which is crucial for traffic, and one which contains maintenance data. The performance requirements on the traffic data are severe. This means that

1. The traffic data must be kept in memory at all times.
2. The traffic data must be organized in such a way so that when an external stimuli arrives to the system, all data which is necessary to process that stimuli must be located in a very efficient manner

The maintenance data usually contains a copy of the traffic data, although structured in a different way. Traditionally, the management of traffic data in telecommunications applications has not been performed by a general purpose DBMS, but rather tailor made for each application. One of the design goals for Mnesia was to provide a DBMS where both the requirements on the traffic data as well as the maintenance data are met.

The rest of this paper is organized as follows: Section 2 is an Erlang introduction and section 3 is an overview of Mnesia. Some usage of Mnesia is given in section 4 and some notes of the future and conclusions are given in section 5 and 6.

2 Erlang

Erlang[AVWW95] is a symbolic functional language intended for large real-time applications mainly in the telecommunication area.

Functions may have many "clauses" and the correct one is selected by pattern matching. The visibility is controlled by explicit export declarations in modules.

Processes are lightweight and created explicitly. They communicate with asynchronous message passing. Messages are received selectively using pattern matching. When running on UNIX there is only one UNIX process namely the Erlang node. This node has its own internal process handling, so the Erlang processes should not be mistaken for UNIX processes.

Erlang nodes can be run on different types of machines and operating systems and the Erlang processes can communicate across a network with processes on other nodes. This does not influence the program writing, except if the message passing time is critical and the physical transport medium is slow. No extra protocol specifications are necessary. The

message passing is competitive with other network communication methods [Wik94].

There are two kinds of support for error recovery in the language: exceptions and special exit messages propagated to linked processes when a process dies. Those messages can be caught by the other processes to activate some recovery plan, for instance restart the dying process.

ERLANG programs communicate with hardware and with programs written in other languages by *ports*. The ports behave similar to processes in that they can receive messages from processes and send messages back to them. The difference from processes are that ports are connected either to non-ERLANG software or to hardware.

Example 1 The `append` function which returns the two argument lists appended is in ERLANG:

```
append([H|T], Z) -> [H|append(T,Z)];
append([], X) -> X.
```

□

3 Mnesia

3.1 Features

The DBMS Mnesia provides:

a. storage of full ERLANG variant length terms:
   (i) primary memory for fast access (with optional disk backup)
   (ii) disk-only for large data sets

b. a logic query language

c. a fast program interface for very simple queries

d. distributed transactions and query evaluation

e. crash recovery

f. replication of data on separate nodes

g. location transparency: applications are written without knowledge where and how the tables are located

h. run-time schema alteration

Those features are for Mnesia's fulfillment of the requirements given in the introduction. The mapping from the requirements to the features could be pictured in a table:

Mnesia feature	Requirement 1	2	3	4	5	6
a(i)	x					x
a(ii)						x
b		x				
c	x					
d			x			
e				x		
f	x		x	x	x	
g					x	
h					x	

Requirement and feature mapping

The feature list above is to be seen as an options list for the Mnesia user. For data which is access time critical but where it need not survive a crash, the user selects primary memory storage only. It is possible to dump the contents of a primary memory table to disk at regular intervals. Data which must survive a crash, and have high requirements on access time, can be confugured in such a way that the data is always kept in RAM and all write operations on the data are logged to disc. The system will then at regular intervals checkpoint the data and truncate the log. Data which is not performance critical, can be kept on disc only.

An advanced feature of Mnesia is that all system activities that are performed, such as checkpointing, dumping tables, schema reconfigurations, e.t.c never stop the system. So for example, although a table is being copied to a remote node, the table is still available for write operations to all applications.

3.2 Data model and schema

Data is organized in tables. A table has a set of properties, for example:

- on which node it is located
- where it is replicated (= a list of zero or more nodes)
- storage (RAM, RAM with Disc-copy or Disc-only)
- if the system should maintain any indices

The table definitions and the table properties are collected in a schema which may be altered in run-time.

Each individual field in a row in a table, may be arbitrarily complex and contain compound ERLANG terms including object code, lambda expressions and regular data. This is one of the features which makes Mnesia suitable for many telecommunications problem where the single most important requirement on the DBMS is that an individual lookup is very fast. This property makes data modeling very flexible and it allows for a data model where all the data which is necessary for i.e. a call in a telephony switch, to be bundled in a single record.

3.3 Concurrency control and crash recovery

Operations on a database are grouped into transactions which are distributed over the network. All completed transactions are logged as well as the 'dirty' operations where a time-critical application can bypass the transaction system.

The lock manager uses a multitude of traditional techniques. Locking is dynamic, and each lock is acquired by a transaction when needed. Regular two-phase locking [EGLT76] is used and deadlock prevention is traditional wait-die [RSL78]. The time stamps for the wait-die algorithm are acquired by Lamport clocks [Lam78] maintained by the transaction manager on each node. When a transaction is restarted, its Lamport clock is maintained, thus

making Mnesia live lock free as well. The lock manager also implements multi granularity locking [GLPT75].

Traditional two-phase commit [Gre78] is used by the transaction manager when a transaction is finished.

Since Mnesia is running on top of distributed ERLANG the implementation is greatly simplified. In a distributed application there are separate ERLANG nodes running on (usually) different machines. ERLANG takes care of the communication between processes possibly on separate nodes transparently. Processes and nodes can easily be started, supervised and stopped by processes on other nodes. This makes lots of communication implementation problems disappear for Mnesia as well as for applications.

Mnesia has (at least) one process on each participating node. Those processes takes care of updates, transactions, emergency reconfiguration at node failure as well as normal maintenance reconfigurations.

The user API to the transaction system is straightforward and easy to use. The programmer presents the transaction system with a lambda expression and a closure, which is the executed by the transaction system.

3.4 Queries

The query construction is integrated with the ERLANG language by list comprehensions. This removes the impedance mismatch that a solution like Embedded SQL or some embedded kind of Datalog[Ull89] would give with a functional language like ERLANG. The use of list comprehension in connection with relational DBMS has been known for a long time[Bre88].

Example 2 The ERLANG list comprehension:

```
[E.name ||
        E <- table(employee),
        D <- table(department),
        E.name = D.boss,
        E.sex = female]
```

extracts a list of the names of all female bosses in a database. This is equivalent with the Datalog expression:

```
employee(Name,_,_,female,_,_),
department(_,_,_,Name)
```

□

We think that the straight forward translation between List Comprehension and Datalog makes a nice and clean connection.

A logical query language is chosen by two reasons. The first is that one of the authors has a background in logic programming and the second is that we will need to make an SQL interface in the future. We think that with a logical language as target the SQL translation will be somewhat simpler.

Cursors are available for the users. If the node of the querying user process is running on a multi-cpu the database query could sometimes be parallelized and will always run in parallel with the user process.

3.5 About the Implementation

All of Mnesia is written in ERLANG except from a special key-value dictionary using linear hashing [Lar88]. This dictionary was added to the ERLANG implementation to get fast key-lookups.

The majority of our present users queries are non-recursive. Therefore we have chosen to separate the recursive and the non-recursive parts of the queries. In the first version we make the non-recursive query evaluation efficient and in a later version the recursion will be more efficient than today.

Minimal cycles in the query flow graph are collapsed into one node giving a non-cyclic graph corresponding to a non-recursive query with explicit recursion operators. This non-recursive query is optimized and compiled using standard methods like partial evaluation and goal reordering guided by statistics about the database. Note that there is no optimization of recursive parts (yet). They are simply put last in the query.

The query is evaluated by the relational DBMS technique operators[Gra93]. The recursive parts are evaluated by SLG [WC93, WCS93, CW93] resolution. This is currently implemented by a naive straight-forward interpreter.

The operator technique maps extremely well onto the ERLANG processes and messages — as a spin-off effect queries are automatically parallelized when run on a multi-cpu computer.

4 Projects

Mnesia is currently used in several development projects within Ericsson. Some of these projects are small scale prototype projects and some are large scale product projects. Mnesia is being used for both traffic data as well as maintenance data in various ways and the applications include pure switch controllers, Intelligent Network controllers as well as office/telephony/internet applications.

5 The Future

Primarily we will evaluate Mnesia with the applications that are now being written. We will compare the performance with commercial DBMS as well with modern research DBMS. Preliminary performance measurements are promising.

Mnesia has no consistency checking at updates today. We will therefore add integrity constraint checking in a near future.

In the area of query processing there are two obvious continuations namely making an *efficient* SLG implementation and to optimize recursive queries.

We will try other transaction and locking algorithms, especially real-time algorithms.

There is a need by some of our users of an SQL-interface. This will be provided and probably also an ODBMC interface. This will enable external PC-applications to access the DBMS.

Another problem which has not been addressed is the problem of partitioned networks. If the network between two Mnesia nodes fail, but the nodes themselves continue to operate, we have a problematic situation.

6 Experiences and Conclusions

By combining wellknown algorithms with the symbolic high level language ERLANG it is possible to write a full industrial DBMS with only a fraction of the manpower normally required. We have by this also obtained a good platform for research and got some real world applications for benchmarking.

References

[AVWW95] Joe Armstrong, Robert Virding, Claes Wikström, and Mike Williams. *Concurrent Programming in ERLANG*. Prentice Hall, second edition, 1995.

[Bre88] P T Breuer. Applicative query languages. Technical report, Cambridge University Engineering Dept, 1988.

[CW93] A. W. Chen and D. S. Warren. Query evaluation under the well-founded semantics. In *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys.*, page 168, Washington, DC, May 1993.

[EGLT76] K.P Eswaran, J.N. Grey, R.A. Lorie, and I.L. Traiger. The notions of consistence and predicate locks in a database system. *Communications of ACM*, 19(11):624–633, November 1976.

[GLPT75] J.N. Grey, R.A. Lorie, G.R. Putzolo, and I.L. Traiger. Granularity of locks and degrees of consistency in a shared database. Technical Report Research report RJ1654,, IBM, September 1975.

[Gra93] Goetz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, June 1993.

[Gre78] J.N. Grey. Notes on database operating system: An advanced cource. *Lecture notes in Computer Science, Springer Verlag, Berlin*, 1(60):393–481, 1978.

[Lam78] L Lamport. Time, clocks and the ordering of events i a distributed system. *ACM Transactions on Programming Languages and Systems*, 21(1):558–565, July 1978.

[Lar88] P-Å Larson. Dynamic hash tables. *Communications of the ACM*, 31(4), 1988.

[RSL78] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis. System level concurrency control for distributed databases. *ACM Transactions on Database Systems*, 3(2):178–198, June 1978.

[Ull89] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 2. Computer Science Press, 1989. ISBN 0-7167-8162-X.

[WC93] David S Warren and Weidong Chen. Towards effective evaluation of general logic programs. Technical report, Computer Science Department (SUNY) at Stony Brook, 1993.

[WCS93] D. S. Warren, W. Chen, and T. Swift. Efficient computation of queries under the well-founded semantics. Technical Report 93-CSE-33, Southern Methodist University, 1993.

[Wik94] Claes Wikström. Distributed computing in Erlang. In *First International symposium on Parallel Symbolic computation*, Sep 1994.