

Evaluation of Database Management Systems for Erlang

Author:
Emil Hellman,
Software Engineering and Management,
IT University of Göteborg,
it3heem@ituniv.se

Academic Supervisor:
William E. Sullivan,
IT University of Gothenburg

Industrial Supervisor:
Francesco Cesarini,
Erlang Training and Consulting Ltd.

Abstract

Erlang/OTP's DBMS Mnesia is lacking in several important areas to consider when implementing very large databases with massive scalability requirements. This article reveals the result from a study examining what Erlang developers consider important aspects of DBMSs and an AHP evaluation on four mature open source DBMSs based on those criteria. AHP is suggested as good method to evaluate DBMSs for Erlang projects. The criteria used in this evaluation were derived from a survey on sent to the Erlang community. It should therefore be noted that which DBMS to use in Erlang projects should also be determined by the project's and the software's own specific criteria.

Keywords: Erlang, DBMS, AHP

Introduction

Erlang is a general-purpose programming language and runtime environment. Erlang has built-in support for concurrency, distribution and fault tolerance [1], having been specifically designed for constructing high reliability telecommunications systems [2].

Erlang's suitability for use in telecom system is supported by the research of Nyström, Trinder and King [3]. According to Nyström et al. Erlang telecom systems are able to meet the specific scalability, resilience, fault-tolerance and dynamic adaptability requirements. In fact, a Dispatch Call Controller built as a basis for analysis presented in [4] showed an increase by a factor of 14.55 in throughput when 16 processing elements were added which shows that Erlang exhibit very good scalability.

However, developing software system in Erlang isn't limited to the telecom industries. There is a growing amount of work being done with the language in AI, banking, track and trace, and web based systems. As such, Erlang's solid scalability, robustness and soft real-time capabilities either needs to be complemented by software which meet the requirements of these new areas of application, or Erlang needs to be extended to support those requirements.

One of the aspects of these new areas of application is the need for large data storage. Erlang features a DBMS named Mnesia which could be used to store the intricate relations between database elements. According to the Erlang website, Mnesia is foremost a memory-resident database [1]. It is suited for distributed systems and has good transaction control capabilities. However, it is rather limited in its storage capabilities. Mnesia tables can only store 4 GB in the 32-bit implementation. Therefore, if a system has larger storage requirements than this, like the track and trace systems which continually grow over time, Mnesia is not a valid choice. 4 GB of data is not a large amount of data, so Erlang developers must find a way to meet these system requirements.

One way is to utilize other already existing DBMSs that are not limited in their storage capabilities. Which DBMS to choose, however, is highly dependent on the system being built. Therefore, a method of evaluation needs to be developed. What criteria to use, how to weigh them, and how to solve inter criteria dependencies needs to be addressed. This thesis presents and applies such a method, specifically aimed at judging the suitability of a DBMS for use with Erlang in building very large scalable database systems. The evaluated database management systems are not supposed to replace Mnesia, but instead complement it in situations where Mnesia is an unsuitable choice.

The problem is which of the open source DBMSs; MySQL, PostgreSQL, Berkeley DB, Ingres is most suited for use with Erlang in building highly scalable systems with no impeding upper limit in its storage capabilities. The sub problems of what criteria to use and how to weigh them and address possible inter dependencies between them will also be discussed.

Scalability

This paper brings up the issue of scalability in database management systems, in particular, their ability to store massive amounts of data. However, there is "no adequate, commonly accepted definition of scalability" according to [5]. A number of definitions flourish in research articles:

“Scalability is a property which exhibits performance linearly proportional to the number of processors employed.”[5]

and

“Scalability means not just the ability to operate, but to operate efficiently and with adequate quality of service, over the given range of configurations. Increased capacity should be in proportion to the cost, and quality of service should be maintained.” [6]

The definition of scalability below is, however, slightly more comprehensive:

“... scalability indicates the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added.” [7]

With this definition scalability can come in in two flavors, scale-up and scale-out. Scaling out means to add additional computational nodes to an existing system and thereby making it able to handle problem of a larger size. An example of scaling out would be to add a computer to a cluster or replicate a database to an additional node. Scaling up a system refers to the activity of increasing the resources available at a specific node. For instance, increasing the processor speed, hard drive speed or adding additional memory to the node. Generally speaking, scaling out offers a cheap way to increase the total throughput of a system, as this scalability flavor can simply involve buying a standard off the shelf computer and plugging it into your current cluster. In the context of databases there are additional elements to consider.

For write intensive database systems the ability to scale out is limited. In fact Gray et al. shows in [8] that replicated databases systems are detrimental to performance. This is because the size of a transaction increases proportionally to the number of replicas in the system. However, [9] has recently suggested a middleware replication tool that in certain situation increases the performance of a cluster even with mostly update transactions. This is due to the removal of computational overhead. This mostly concerns the performance scalability of a system. However, this is simply one way of viewing scalability. Performance can in the common case be solved by throwing money at the problem (by scaling up with better harddrives, faster CPU, and more RAM).

Applying the definition of scalability to the data storage capabilities of a system we can view scalability as a system's capacity to store increased amounts of data with added physical storage capabilities. That is, if we increase the amount of physical storage available, can the system use that storage? As previously indicated, Mnesia has its limitations in this respect.

Alternatives

The databases evaluated in this study is Ingres 2006, PostgreSQL 8.1, MySQL 5.1 and Berkeley DB 4.4. These four DBMSs are among the most mature and well known. All of them are also open source which is a good match for Erlang, which is also available as open source. Those are the essential reasons that these DBMSs have been included in the evaluation.

Ingres 2006 is a enterprise grade open source relational database management system. It is available under the GNU General Public License version 2. There is also a commercial license available if Ingres client libraries are to be bundled in non-GPL software [10].

PostgreSQL 8.1 is advertised as the world's most advanced open souce database. It is available under the BSD license [11].

MySQL is a widely used DBMS for web applications under the Linux, Apache, MySQL and PHP (LAMP) configuration. There is a dual licensing scheme for MySQL. For open source projects using the GPL license, MySQL is free to use under the terms of the GPL license. For distributions of MySQL with products that does not provide its source code under the GPL, a commercial license is available [12].

Berkeley DB is a popular choice for embedded databases. It is available under two licensces; the Sleepycat Public License and the Sleepycat Commercial License. The latter should be used for non-open source applications which are distributed two third parties [13].

Method

Two general methods of evaluation have been considered for use as a basis for the evaluation, The Logic Scoring of Preference and the Analytic Hierarchy Process. A description of each follows and motivation of the final choice of method is given in the following few sections.

LSP

The Logic Scoring of Preference (LSP) is an evaluation method developed by Jozo J. Dujmović. It is a ”general quantitative decision method for evaluation, comparison, and selection of complex ... software systems” [14]. The main advantage for this method is that it is capable of handling dependencies between different criteria. For instance, if a certain attribute is desired, but can be compensated for by other system attributes, it is possible to model this mathematically with the method. Sequences of operators can be used on combinations of attributes to produce an accurate score for the systems suitability.

AHP

Another method for evaluation is the Analytic Hierarchy Process (AHP). It was developed by Dr. Thomas Saaty and is more arbitrary than the LSP method. To begin with, the evaluator chooses which criteria are important in the evaluation. Their relative importance (weights) are decided by doing a pairwise comparison of all criteria. For each criteria, the evaluator does a pairwise comparison of the alternatives (in this case the DBMS). The scores for the alternatives are then multiplied by the weight of the criteria. The alternative with the highest total score is the optimal choice given the specific criteria and alternatives.

The main benefit of using this method is its relative simplicity. It is also a good choice when the criteria are hard to formalize, since the pairwise comparison does not need to be based on numbers. There is no absolute value involved in contrast to the LSP method, which relies on quantifiable data. Instead, the AHP method simply relies on the relation between criteria in conjunction with the relation between the alternatives for each of the criterion.

The comparison scale used is one proposed by Saaty in [15]. The scale has nine levels as shown in Table 1.

Intensity	Definition
1	Equal importance
2	Weak
3	Moderate importance
4	Moderate plus
5	Strong importance
6	Strong plus
7	Very strong or demonstrated importance
8	Very, very strong
9	Extreme importance

Table 1: Scale of relative importance according to [15]

There are several versions of AHP and the difference between them lies in how consistency is achieved. Two main methods exist according to [16] the eigenvalue approach proposed in [15, 17] and “methods minimizing the distance between the user-defined matrix and the nearest consistent matrix” [16], such as the calculating the geometric mean.

Method of Choice

Since the criteria found in the study were hard to quantify (replication abilities for instance) the method chosen for the comparison was AHP. The ease of use of the method also makes future modification and analysis simpler. While LSP is a good evaluation technique when the criteria of the evaluation have complex interdependencies this is not the case in this study.

Even if this was the case, the AHP method can be placed on a higher level where a criterion takes both of the two related criteria into account. The eigenvalue approach was used to achieve consistency in the evaluations.

Data Collection

The collection and selection of criteria to use in the evaluation was conducted in two steps. First, a number of criteria were found through various articles (For instance, [6, 18]). These were then evaluated through a survey sent to the Erlang Mailing list in order to verify that Erlang developers considered the presented criteria valid for a basis of evaluation of DBMSs to use with Erlang. In addition, key Erlang developer with experience working with database systems were identified and specifically asked to answer the questions in the survey. A side effect of targeting specific developers were that the answers to the survey became more elaborate and relevant. Because of this a few additional criteria arose which were included in the evaluation. The results from the survey were used as a basis for assigning the weights of each criteria. This was done through a combination of looking at the number of times a specific criteria was indicated as an important feature of a DBMS and by analyzing what the survey participants considered to be the major flaws of Mnesia.

Several questions were asked in the survey, however the most significant ones were:

- Has there been times when Mnesia was not scalable enough for your requirements? If so, in what was Mnesia lacking?
- What do you consider the three most important technical characteristics of a DBMS?
 - Safety mechanisms (failover on node crashes, crash recovery mechanisms etc.)
 - Replication abilities of the DBMS
 - Ability to create Views
 - Number of simultaneous users supported
 - Ability to handle large data quantities
 - DBMS interface to Erlang
 - Other attributes, Please Specify

It was mainly on these two questions that the criteria and their weights were based. The entire survey can be found in Appendix A.

The scores of the systems being evaluated were derived from the information available from the DBMS vendors [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30].

Results

In the following sections the result from the survey is presented followed by a calculation of relevant criteria weights. Following this the evaluated DBMSs are compared for each criteria and a summary is made.

Survey

Initially, the criterion listed in question 3 of the survey (see Appendix A) were considered to be important for any potential candidate. Among these, the ability to handle large amounts of data was stated as the major flaw of Mnesia. Mnesia's replication abilities were also considered flawed by some of the survey participants:

“Mnesia replication ... is not reliable enough. Once partitioned, Mnesia node does not reconnect automatically (at least not in R9.x which we're using), thus rendering it inadequate”

Replicating a database is important since it allows the system to scale better. At least read queries can be shared between several replicas, thus increasing throughput.

Replication and data storage capabilities were not only the major flaws of Mnesia, but also indicated as two of the most important features of a DBMS. These factors attest to the importance of the two criteria, and in Table 2, this is signified by their high relative importance compared to the other criterion. Data storage capabilities were given a slightly higher degree of importance due to it being indicated by more participants as a flaw in Mnesia.

Safety mechanisms such as fail over on node crashes was the third criteria mentioned in the survey that the participants thought was important. The reasoning behind this could stem from Erlang systems superb availability capabilities and a desire that the DBMS to interact with is as reliable in this aspect.

A fourth criteria that was considered important was the DBMS interface to Erlang. Although not as many people considered this criteria as significant as the above ones, it is likely a deciding factor in if the DBMS will be adopted by the community. Legris et al. suggest that two of the most important factors influencing technology adoption are perceived ease of use and perceived usefulness [31]. A proper Erlang interface would as such increase the likelihood of the DBMS to be adopted by the community.

Developing one from scratch is not a trivial challenge and a likely deterrent for potential users.

The other criterion presented in the study was not considered important by most of the survey participants. Instead, the result showed that Erlang developers had some additional requirements, that were not mentioned in the survey, which they regarded as important.

Of the additional requirements, the ability to maintain logical constraints was advocated by one of the survey participants:

“When the project grows and the number of developers increases it is important that the database is capable of maintaining the logical constraints of the model. Mnesia is completely lacking in this respect.”

Although this criteria was only mentioned by one of the survey participants it was added to the criteria used in the evaluation in Table 2. The reason behind this is that the DBMSs should complement Mnesia's flaws. When creating a table in Mnesia there is no way of imposing logical constraints, such as ensuring uniqueness of a table attribute (other than the key of sets and ordered sets) [32]. As such the constraints must be implemented in the programming logic, which is not always desirable when the database system interacts with several other systems.

Familiarity with the technology, performance, adherence to standards, and tool availability were other criteria that was brought up by the survey participants. These were not included in Table 2 of evaluation criteria.

While familiarity with the technology is an important issue to consider when choosing a DBMS for a project, it is not very helpful when identifying a suitable complement to Mnesia. In the Erlang community, developers are bound to have worked with a wide range of DBMSs. As such specifying the familiarity with the technologies for the entire Erlang community is not an essential point of comparison since a specific organization will not necessarily concur with that assessment. It is likely that an organization has a high degree of familiarity with entirely other DBMSs than the average of the Erlang community.

The performance of DBMSs was an important issue for many of the Erlang developers that participated in the survey. However, performance is highly dependent on how (well) a DBMS has been configured for a specific system. Additionally, the metric is tightly coupled with the use of the underlying hardware. Because of this, the criteria was discarded from the comparison. Like with the familiarity with the technology, this should be evaluated on a per system basis, and not on a general level as in this evaluation.

The last two criteria mentioned (Adherence to

standards and Tool Availability) was not included in the evaluation because too few survey participants considered them important. Also adherence to standards can be viewed as a part of the Erlang interface criteria. If the interface to the DBMS is a clearly specified and widely used standard it will be easier to interact with the DBMS.

Table 2 shows the relative importance of the evaluation criteria based on the answers to the survey.

Criterion	Safety	Large Data	Replication	Maint. Logical Constr.	Erlang Interface	Eigen Value
Safety	1	0.2	0.2	3	2	0.1622
Large Data	5	1	2	7	5	0.4054
Replication	5	0.5	1	6	4	0.3243
Maint. Logical Constr.	0.33	0.14	0.17	1	0.33	0.0270
Erlang Interface	0.5	0.2	0.25	3	1	0.0811

Table 2: Relative importance of the evaluation criterion and the resulting Eigen values after consistency calculations.

The eigenvalues for the criterion matrix when consistency has been achieved are the resulting weights for the criterion. Ability to store large amounts of data is the most important criteria consuming about 41% of the total followed by Replication capabilities at 32%. Safety mechanism weighs 16% and the ability to maintain logical constraints and the Erlang interface has 3% and 8% respectively.

Beside indicating the most important criteria for the evaluation the survey also showed that many Erlang developers still used Mnesia for projects needing large databases. At least in one case this has created a major overhead for maintenance.

Evaluation of Alternatives

For each of the criterion above, an evaluation is made of the alternative DBMSs. The basis for the scores is derived from [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30].

Safety Mechanisms

Ingres

Ingres have extensive recovery features according to [19]. It has the ability to recover an entire database based on a backup copy, but it can also recover only selected tables in the case that Ingres “marks less than the whole database as physically inconsistent” [19]. Backups can be made as

snapshots or using checkpointing. In combination with Ingres journaling system checkpointing can be used to restore data up to the last journaled transaction.

By using Ingres Replicator Option, one can configure the system so that if one node fails the users switch to another. If a crash or other event causes replicas to become inconsistent, the Ingres Replicator Option has methods to bridge the difference that works with the normal procedures for recovery [20].

PostgreSQL

Two main methods can be used to backup a PostgreSQL database. An SQL dump can be created with tools shipped with PostgreSQL. The more advanced backup feature is the On-line backup which works in conjunction with the write ahead logs that this DBMS creates. The write ahead logs details all changes made to the database's data files and by replaying these changes the system can recover from crashes [21].

A hot standby system can be setup by distributing the write ahead logs to another node. The on-line backup method can not do partial recoveries of a database. It can only restore an entire database cluster. The PostgreSQL side project, Slony-I, support switch- and failover [22] and PgCluster is also an option although its setup seems a bit more advanced [23].

MySQL

MySQL, like PostgreSQL, has a tool for creating a dump of the database which can be used as a backup. There is also another way of backing up the database. This method, however, requires that the relevant tables are first locked with at least a read lock and then flushed to ensure that indexes are written to disc. With proper configuration, a MySQL database can use binary log files to recover from a specified point in time to any subsequent moment.

With master/slave replication activated it is possible to setup a script that when the master fails promotes a slave to master. With the NDB storage engine it is possible to configure the system for failover [24].

Berkeley DB

For Berkeley DB, logging of operations is left as a responsibility for the application. However, it does provide tools for auto generation of logging functionality. It is possible to create checkpoints after which previous log files can be removed. Consistent on-line backups can be taken, but there it is not possible to know the exact instance at which they are created. It is possible to setup a failover mechanism with Berkeley DB, it must however involve the application using the database [25, 26, 27].

Comparison

Table 3 shows the comparison between the evaluated DBMSs on their safety mechanisms. Ingres was deemed moderately better than the other DBMSs due to its ability to do partial recoveries. Berkeley DB was considered slightly worse than the others because of its reliance on the applications using it.

Safety Mechanisms	Ingres	PostgreSQL	MySQL	Berkeley DB	Eigen Value
Ingres	1	3	3	4	0.5158
PostgreSQL	0.33	1	1	2	0.1894
MySQL	0.33	1	1	2	0.1894
Berkeley DB	0.25	0.5	0.5	1	0.1054

Table 3: Safety Mechanism comparison matrix with eigenvalues

Ability to Store Large Sets of Data

Ingres

The database is not limited in its storage capability other than by the available physical storage of the hardware. However, tables while they are not limited do have performance penalties that are introduced when the table rows grow beyond the page size specified for the table. The maximum row size before this performance penalty occurs is theoretically 256 GB. With such a large row however, selecting the row will probably have performance problems in any case.

Ingres can only have 67108863 number of tables. Each of these can at the most store 4,294,690,816 rows in a table unless the table is partitioned [28, 19].

PostgreSQL

PostgreSQL has no maximum limit on the database size (as long as the storage medium is not out of space). It is however limited to a table size of 32 TB (once properly configured). The rows in a table is limited to 1.6 TB. Table size is not limited by the OS because tables are fragmented into several 1GB files.

Both the number of tables in a database and the number of rows in a table is unlimited¹ [21].

MySQL

The storage limits on MySQL depends on which storage

¹ Obviously discounting possible hardware, OS, and file system limitations.

engine is used. With the MyISAM storage engine the table's size is most likely limited by the OS and underlying file system since the limit is 65536 TB. For instance, on a Linux 2.4+ computer using ext3 file system the file size is limited to 4 TB. Using the InnoDB storage engine several a table can be spread over several files making up a tablespace. A InnoDB tablespace is limited to 64 TB.

MyISAM tables have a maximum row size of 64KB however this limit does not include the size of BLOBs and TEXT types. InnoDB tables have a row size limit of 4GB, however this includes any BLOBs or TEXT type columns. Without VARCHAR, TEXT, or BLOBs the row size is 8000 bytes (about half a database page).

MySQL has no limits on the number of tables in a database nor the number of rows in a table, besides those imposed by the underlying hardware [24].

Berkeley DB

A Berkeley DB database is limited to 256 TB in size which naturally is also the maximum size of a single table since a database in Berkeley DB corresponds to a SQL table. A single record (key, value) Records up to 4GB and tables up to 256TB.

The maximum number of logical records that can be stored in the database is 4,294,967,295 [26, 27].

Comparison

PostgreSQL is the superior database in this comparison due to the possibility to have very large tables and not being limited by the OS. Table 4 reveals that Ingres is deemed slightly better than MySQL. The reason for that is MySQL's small row size. Berkeley DB draws the short straw since it's limit on the record sizes.

Storage Capabilities	Ingres	PostgreSQL	MySQL	Berkeley DB	Eigen Value
Ingres	1	0.5	2	3	0.2739
PostgreSQL	2	1	3	4	0.4620
MySQL	0.5	0.33	1	3	0.1780
Berkeley DB	0.33	0.25	0.33	1	0.0862

Table 4: Comparison matrix of the DBMSs' storage capabilities.

Replication Capabilities

Ingres

Replication is available through the Ingres Replicator Option. It is possible to set up a system with different databases with the replicator option, however some limits may apply. Replication can be set up to replicate data; from one database to another, from one database to many others, in both directions between two databases, or in many directions between many databases. It is also possible to select what data should be replicated from an entire database to or subsets of tables columns or rows [20].

Ingres can also be setup as a cluster. This places some additional requirements on the underlying system and some features are not available [30].

PostgreSQL

Slony-I supplies replication capabilities to PostgreSQL servers. However, it has some limitations:

- It does not automatically propagate schema changes
- It does not have any ability to replicate large objects [22].

An alternative to Slony-I is PgCluster. It provides replication and load balancing. Failure of one of its “cluster” nodes does not disrupt system to handle other queries [23].

MySQL

MySQL provides one way replication. It can be either row based or statement based. One limitation of the MySQL's replication capabilities is that a master server cannot replicate to slaves using older versions of MySQL in the case that the master utilize a feature not present in the slave's MySQL version.

With the NDB storage engine it is possible to setup a database cluster where the failure of one machine does not effect the system as a whole. It disrupts the current transaction, but the system will still be able to function [24].

Berkeley DB

A Master/Slave replication systems can be setup with Berkeley DB, however it has to involve the application using it.

There are no clustering capabilities innate to Berkeley DB [25, 26, 27].

Comparison

Ingres ability to replicate at a finer granularity than the other DBMSs makes it slightly superior in the aspect of replication. MySQL and PostgreSQL are roughly evenly matched. PostgreSQL seems slightly better at replication, but MySQL's clustering capabilities appear more mature. Berkeley DB's dependency on the application using it is why it is considered inferior. Table 5 shows the comparison between the DBMSs. The Eigenvalue column indicates their scores for replication.

Replication	Ingres	PostgreSQL	MySQL	Berkeley DB	Eigen Value
Ingres	1	3	3	4	0.5072
PostgreSQL	0.33	1	1	3	0.2038
MySQL	0.33	0.1	1	3	0.2038
Berkeley DB	0.25	0.33	0.33	1	0.0830

Table 5: Comparison matrix of the replication capabilities of the DBMSs.

Maintenance of Logical Constraints

Ingres

Ingres supports three kinds of constraints: Unique, Referential, and Check constraints. There is also integrity objects which checks update requests on a database before they take place.

“Rules” allow the user to define specific database events to react to, for instance the insertion of specific values into a table. Rules are always triggered after an update, insert or delete [19].

PostgreSQL

Like Ingres PostgreSQL supports unique, check and foreign key referential constraints. It is possible to create triggers of two types; row triggers, which can execute after or before a row is effected, and statement triggers, which similarly can execute either before or after the statement has effected the database.

In PostgreSQL there is also a rules system which can be used to rewrite queries to the database. The rules system exist in addition to the trigger system and they can often be used to accomplish the same goals, which to use is dictated by how the database is used [21]

MySQL

Constraints in MySQL is not as strict as in Ingres and PostgreSQL. By default MySQL may force an illegal value to the closest possible legal value. It is, however, possible to

configure the DBMS to generate errors and abort instead. If InnoDB is used as the storage engine the statement is rolled back when an error occurs. For non-transactional storage engines however, the execution of a statement simply stops at the row the error occurred. Any changes made before reaching that row are not aborted. [24].

Berkeley DB

Aside from keys and indexes it is not possible to set any constraints on a Berkeley DB database [25] unless of course the application is involved. There is no trigger functionality in the database [27].

Comparison

PostgreSQL and Ingres have similar capabilities for maintaining logical constraints. MySQL, however, is judged as slightly worse than the others due to inappropriate default behavior. Because Berkeley DB lacks many of the more refined mechanisms for maintaining logical constraints it is deemed inferior to the other DBMSs. Table 6 details the comparison matrix for the DBMSs abilities to maintain logical constraints.

Maint. Logical Constraints	Ingres	PostgreSQL	MySQL	Berkeley DB	Eigen Value
Ingres	1	1	2	7	0.3699
PostgreSQL	1	1	2	7	0.3699
MySQL	0.5	0.5	1	6	0.2137
Berkeley DB	0.14	0.14	0.17	1	0.0465

Table 6: Comparison matrix for maintaining logical constraints.

Erlang Interface

All of the DBMSs provides C interfaces which can be used for developing drivers to communicate with them.

Ingres

There is an ODBC application shipped with Erlang/OTP that can be used to communicate with any DBMS that supports a ODBC driver of version 3.0. Beyond this Erlang application there is no publicly available Erlang driver for Ingres.

PostgreSQL

There are no publicly available Erlang drivers for PostgreSQL beyond the ODBC application. Erlang Training and Consulting have developed a driver application for

PostgreSQL which can handle multiple connections to a database. It does not support conversion of all of the PostgreSQL data types and does not currently handle database errors in a proper manner. Once these issues have been amended, the driver will be released as open source.

MySQL

Aside from the ODBC application is a basic MySQL module available at <http://www.erlang-projects.org>. However, it has very limited functionality according to its description.

Berkeley DB

There is a Berkeley DB driver available from <http://www.snookles.com/erlang/>. It is designed for Berkeley DB 4.0.14 and lacks support for some features. There is also a Berkeley DB Erlang Term Store which has been developed by Synapse and was presented at the 2004 Erlang User Conference by Per Bergqvist [29]. However, Bets has not been thoroughly examined and is not available as open source.

Comparison

Of the relational DBMSs in the comparison, Ingres is currently the least capable to interact with an Erlang program since the only way of achieving such communication is through ODBC. The PostgreSQL driver by Erlang Training and Consulting is more advanced than the MySQL driver on Erlang projects, and as such PostgreSQL, is deemed slightly better than MySQL. The available interfaces to Berkeley DB is judged as slightly worse than Ingres's mainly because the limitations of the interfaces that were looked at.

Erlang Interface	Ingres	PostgreSQL	MySQL	Berkeley DB	Eigen Value
Ingres	1	0.33	0.5	2	0.1634
PostgreSQL	3	1	3	3	0.4901
MySQL	2	0.33	1	2	0.2310
Berkeley DB	0.5	0.33	0.5	1	0.1155

Table 7: The comparison matrix for the DBMS's Erlang interface.

Summary

In Table 8 the total score is presented for each of the DBMSs with the weights for the criterion taken into account. The percentage is the amount of the total score each of the DBMSs received.

DBMS	Score
Ingres	37%
PostgreSQL	35%
MySQL	19%
Berkeley DB	9%

Table 8: Total score of each DBMS.

The table shows that Ingres would be the best choice based on the criteria used in this study. There is however, not much difference between Ingres and PostgreSQL's scores. An overview of the DBMSs pro's and con's are shown in a table in Appendix B.

Conclusions

The results show that Ingres would be the best to use in conjunction with Erlang based on the criterion in the study found through the survey of the Erlang Community. However, there are obviously a number of other considerations to make before making the final choice. For instance what performance level we desire and also what licensing limitations the database management system.

Performance can often be increased by buying new hardware. Faster hard drives, memory and such will increase a database's performance more than an engine switch, if the flexibility of a high level query language like SQL is a requirement. However, if one knows what queries will be asked, then SQL introduces a lot of unnecessary overhead. Of the DBMSs evaluated, only Berkeley DB does not use SQL. It might be a good choice to use Berkeley DB for high performance or embedded systems, but for databases with large storage requirements, where the kind of queries that need to be executed is unclear, one of the other DBMSs is probably a better choice.

Organizational requirements might influence the choice of DBMS. PostgreSQL has the most liberal licensing scheme, the BSD license, which does not limit the developers to developing open source systems while still being "free". All the other DBMSs impose some kind of requirements for developing software unless the source code is distributed under the GPL.

If, however, the main requirements on the DBMS for an Erlang project is the ones used as criteria above, then Ingres is the best choice. The developers would have to use the Erlang ODBC driver to communicate with the DBMS (which is known to be troublesome) unless they created their own driver, but other than this, Ingres has no evident flaws. This might not be that surprising, since the DBMS have 30 years of development behind it.

On a final note, Erlang users are not giving up the hope of using Mnesia for very large databases. As a comment left by one of the survey participants who had

been confronted with situations where Mnesia wasn't scalable enough, his alternative was

"We still use Mnesia, with care."

This was in line with what another participant stated, namely that in order to not complicate the design,

"One of our application utilizes MNESIA to store about 200,000,000 rows/records. The database consists of two logical tables: one set of 512 fragments is used for data, and another consisting of 128 fragments is used to store indexes."

It is this type of usage which has taken Mnesia's abilities to what it is today, and will hopefully push it towards the capacities of the open source databases evaluated in this paper.

References

- [1] Erlang, <http://www.erlang.org>, April 17th 2006.
- [2] J. Armstrong, R. Viriding, C. Wikström, and M. Williams, *Concurrent Programming in Erlang*, Prentice Hall, 2nd edition, 1996.
- [3] J.H. Nyström, P.W. Trinder and D.J. King, Are High-level Languages suitable for Robust Telecoms Software? *Proceedings of the 24th International Conference, SAFECOMP 2005*, eds. R. Winther, B.A. Gran and G. Dahll LNCS 3688, Computer Safety, Reliability, and Security, Springer-Verlag, 2005
- [4] Nyström, J. H., Trinder, P. W., and King, D. J. 2003. Evaluating distributed functional languages for telecommunications software. In *Proceedings of the 2003 ACM SIGPLAN Workshop on Erlang* (Uppsala, Sweden, August 29 - 29, 2003). ERLANG '03. ACM Press, New York, NY, 1-7. DOI=<http://doi.acm.org/10.1145/940880.940881>
- [5] Sun X. and Rove D. T, Scalability of Parallel Algorithm-Machine Combinations, *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 6, June 1994.
- [6] P. Jogalekar and M. Woodside, Evaluating the Scalability of Distributed Systems, *IEEE Transactions on Parallel and Distributed Systems*, vol.11, no. 6 pp. 589-603, 2000.
- [7] Wikipedia, *Scalability*, <http://en.wikipedia.org/wiki/Scalability>, April 10th 2006.
- [8] Gray, J., Helland, P., O'Neil, P., and Shasha, D. 1996. The dangers of replication and a solution. In *Proceedings of the 1996 ACM SIGMOD international Conference on Management of Data* (Montreal, Quebec, Canada, June 04 - 06, 1996). J. Widom, Ed.

- SIGMOD '96. ACM Press, New York, NY, 173-182. DOI= <http://doi.acm.org/10.1145/233269.233330>.
- [9] R. Jiménez-Peris, M. Patiño-Martínez, B. Kemme, and G. Alonso, Improving the scalability of fault-tolerant database clusters, *IEEE 22nd International Conference on Distributed Computing Systems*, ICDCS'02, Vienna, Austria, pages 477–484, July 2002.
- [10] Ingres Corporation, *Ingres Licensing Page*, http://www.ingres.com/legal/Legal_Licensing.html, May 20th 2006.
- [11] PostgreSQL Global Development Group, *PostgreSQL license*, <http://www.postgresql.org/about/licence>, May 20th 2006.
- [12] MySQL AB, *MySQL Licensing Policy*, <http://www.mysql.com/company/legal/licensing/>, May 20th 2006.
- [13] Sleepycat Software, *Berkeley DB Licensing*, <http://www.sleepycat.com/company/licensing.html>, May 20th 2006.
- [14] J. J. Dujmović, A Method For Evaluation And Selection Of Complex Hardware And Software Systems, *22nd International Computer Measurement Group Conference*, December 10-13, 1996, San Diego, CA, USA, Proceedings, pages 368-378.
- [15] Saaty Th. L., *The analytic hierarchy process*, MacGray-Hill, New York, 1980.
- [16] Ishizaka Alessio, The Advantages of Clusters in AHP, *15th Mini-Euro Conference*, MUDSM 2004.
- [17] Saaty Th. L., A scaling method for priorities in hierarchical structures, *Journal of mathematical psychology* 15, 234-281, 1977.
- [18] Nussbaum, D. and Agarwal, A. 1991. Scalability of parallel machines. *Commun. ACM* 34, 3 (Mar. 1991), 57-61. DOI= <http://doi.acm.org/10.1145/102868.102871>.
- [19] Ingres Corporation, *Database Administrator Guide*, <http://opensource.ca.com/projects/ingres/documents/product/Ingres%202006%20Documentation/dba/download>, May 15th 2006.
- [20] Ingres Corporation, *Ingres Replicator Option User Guide*, <http://opensource.ca.com/projects/ingres/documents/product/Ingres%202006%20Documentation/rep/download>, May 15th 2006.
- [21] PostgreSQL Global Development Group, *PostgreSQL 8.1.3 Documentation*, <http://www.postgresql.org/docs/8.1>, May 15th 2006.
- [22] GBorg development team, *Doing switchover and fail over with Slony-I*, http://gborg.postgresql.org/project/slony1/genpage.php?howto_failover, May 16th 2006.
- [23] PgCluster, *PgCluster*, http://pgcluster.projects.postgresql.org/1_3/index.html, May 16th 2006.
- [24] MySQL AB, *MySQL 5.1 Reference Manual*, <http://dev.mysql.com/doc/mysql/en/index.html>, May 15th 2006.
- [25] Sleepycat Software, *Programmer's Reference Guide*, <http://www.sleepycat.com/docs/ref/toc.html>, May 15th 2006.
- [26] Sleepycat Software, *Berkeley DB Overview*, <http://www.sleepycat.com/products/bdb.html>, May 15th 2006.
- [27] Sleepycat Software, *Getting Started with Berkeley DB for C*, <http://www.sleepycat.com/docs/gsg/C/BerkeleyDB-Core-C-GSG.pdf>, May 15th 2006.
- [28] Dipl.-Ing. Daniel Fallmann, Dipl.-Ing. Helmut Fallmann, Dipl.-Ing. Andreas Pramböck, Horst Reiterer, Dipl.-Ing. Martin Schumacher, Dipl.-Ing. Thomas Steinmaurer, Univ.-Prof. Dr. Roland Wagner, *Comparison of the Enterprise Functionalities of Open Source Database Management Systems*, Herausgeber und Urheber Fabalabs Software GmbH, Honauerstr. 4 A-4020 Linz, 2005.
- [29] Per Bergqvist, Liberating the mobile internet!, Synapse, *Presentation at EUC October 21st*, 2004.
- [30] Ingres Corporation, *Ingres Getting Started for Linux*, <http://opensource.ca.com/projects/ingres/documents/product/Ingres%202006%20Documentation/gettingstartedlinux/download>, May 17th 2006.
- [31] Legris, P., Ingham, J., and Collerette, P. 2003. Why do people use information technology?: a critical review of the technology acceptance model. *Inf. Manage.* 40, 3 (Jan. 2003), 191-204. DOI= [http://dx.doi.org/10.1016/S0378-7206\(01\)00143-4](http://dx.doi.org/10.1016/S0378-7206(01)00143-4)
- [32] Mnesia, <http://www.erlang.org/doc/doc-5.4.13/lib/mnesia-4.2.5/doc/html/index.html>, May 14th 2006.

Appendix A – Survey

1. Has there been times when Mnesia was not scaleable enough for your requirements? If so, in what was Mnesia lacking?
2. If you answered yes on question 1. What option did you use instead of Mnesia?
 - Ingres
 - MySQL
 - PostgreSQL
 - Berkeley DB
 - Oracle
 - Sybase
 - Other (DBMS or non-DBMS solution), Please Specify
3. What do you consider the three most important technical characteristics of the DBMS you chose above (or Mnesia)?
 - Safety mechanisms (failover on node crashes, crash recovery mechanisms etc.)
 - Replication abilities of the DBMS
 - Ability to create Views
 - Number of simultaneous users supported
 - Ability to handle large data quantities
 - DBMS interface to Erlang
 - Other attributes, Please Specify
4. Are there characteristics that can compensate for lack in those three attributes, if so which and why?
5. Are there any non-technical aspects to consider when looking at a DBMS (Experience with specific DBMS or company directives), if so which and why?
6. On a scale from 1(unimportant) to 10 (highly important), how important was the choice of DBMS in the project(s) needing massive scalability compared to other choices (for instance choosing OS, file system, hardware platform and language in development etc.)?
7. Are there any tools that you used which you considered essential to your development efforts (graphical interfaces to the DBMS etc.)?
If so, which or what were they?
8. Would you like to receive a copy of my thesis once it is finished?

Appendix A – Database Summary Table

Criteria Description	Ingres	PostgreSQL	MySQL	Berkeley DB
Safety Mechanisms	<ul style="list-style-type: none"> + Several Backup Options + Fine granularity + Supports Failover + Can attempt to automatically recover from inconsistencies between nodes after crash. 	<ul style="list-style-type: none"> + Several Backup options + Supports failover and switchover - No partial recoveries of a database - External dependencies for failover and switchover (Slony-1, PgCluster) 	<ul style="list-style-type: none"> + Supports failover + Several Backup options - No partial recoveries of a database - Needs external scripting to do failover 	<ul style="list-style-type: none"> - Failover must involve applications - Logging is a responsibility of application
Ability to Store Large Data Sets	<ul style="list-style-type: none"> + Unlimited DB size - Limited number of tables in a DB - Limited number of rows in a table 	<ul style="list-style-type: none"> + Unlimited DB size + Unlimited number of tables + Unlimited number of rows in a table + 32 TB table size limit independent of OS - Table size limited - Row size limited (1.6 TB) 	<ul style="list-style-type: none"> + Unlimited DB size + Unlimited number of tables + Unlimited number of rows in a table - The row size of InnoDB tables is small (Including BLOBs, VARCHAR and TEXT types it is only 4GB, without 8000 bytes) - MyISAM tables limited by OS 	<ul style="list-style-type: none"> - 256 GB size limit of tables and DB - 4 GB record limit
Replication	<ul style="list-style-type: none"> + Possible to select what data to replicate + Clustering capabilities + Several replication modes available 	<ul style="list-style-type: none"> + Several modes of replication + Several options to choose (PgCluster, Slony-1) - Slony-1's limitations 	<ul style="list-style-type: none"> + Multi-master replication possible + Clustering available with NDB storage engine - Some limitations on replication to Dbs with older versions of MySQL. 	<ul style="list-style-type: none"> - Application involvement necessary
Maintaining Logical Constraints	<ul style="list-style-type: none"> + Unique, Check and referential constraints + Integrity objects + Rules which can invoke procedures 	<ul style="list-style-type: none"> + Unique, Check and referential constraints + Row and Statement triggers (which can execute before and after) + Rules system that complements the trigger system 	<ul style="list-style-type: none"> + Has constraints - May force invalid values to valid ones. - Only with transactional storage engines statements are rolled back when a constraint is violated. 	<ul style="list-style-type: none"> - Does not have constraints, application must handle that
Erlang Interface	<ul style="list-style-type: none"> + Erlang ODBC driver - No interface available beside Erlang ODBC driver 	<ul style="list-style-type: none"> + Erlang ODBC driver + ETC has developed a driver for PostgreSQL which can be used instead of ODBC. 	<ul style="list-style-type: none"> + Erlang ODBC driver + A very limited driver for MySQL is available, aside from the ODBC driver 	<ul style="list-style-type: none"> + Berkeley Erlang Term Storage + Driver available for Berkeley 4.0.14