Efficiency of the Embedded Database System for Handling Outage Data*

Michal Krátký, Radim Bača, and Peter Chovanec

Department of Computer Science,
FEECS, VŠB – Technical University of Ostrava,
17. Listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
{michal.kratky, radim.baca, peter.chovanec}@vsb.cz

Abstract. A database of outages in electrical power network is very important for the black-out prevention. This database makes possible the reliability computation that is applied for the maintenance of equipments in power networks. It is necessary to apply a sophisticated storage for an efficient querying of this data, and so efficient reliability computation. In this paper, we apply a new embedded database system called QuickDB for the storage of outage data. We compare the performance of this database system with common DBMS.

Key words: power networks, reliability computation, outage data, integration of outage databases, DBMS, embedded DBMS

1 Introduction

Institutional changes taking place all over the world drastically effect the approach to power supply quality. It is developing towards a purely commercial matter between suppliers and their customers. The supply that does not comply with agreed qualitative parameters will lead to trade disputes and financial settlements. *Undelivered energy*, including its valuation, has arrived on the scene. The two following aspects of supply quality may be considered:

1. *Supply reliability* – relating the availability of electricity in the given location.
2. *Voltage quality* – relating to the purity of characteristics of the voltage waveform, including the absolute level of voltage and frequency.

This document deals with the first aspect in more detail. Worldwide centers of reliability computation[1] provide databases of information about the availability of electronic and non-electronic components and distribution functions for various failure types. They include the result failure rate and we can retrieve information about the producer, operation conditions, etc. These databases are

* This work was supported by the Czech Science Foundation (No. GA102/09/1842) and by the Ministry of Education, Youth and Sports of the Czech Republic (SGS, No. SP2011/193).

[1] For example Alion System Reliability Center, http://src.alionscience.com/

applicable to the availability prediction of complicated systems. However, these databases do not include data about power equipment.

The IEEE standards define a host of reliability indices applied to distribution reliability. IEEE P1366 [21] explains the reliability indices applied to measurement of distribution system reliability, and a way of calculating reliability indices. Although authors introduce a discussion about some factors influencing these indices, reliability parameters for power system equipment are not depicted. The Canadian Electrical Association[2] introduces a collection of reliability parameters for power system equipment. This is useful for North America; however, it is almost impossible to compare conditions and equipment in North America and Central Europe.

It is necessary to observe failures and outages in the transmission and distribution of electrical energy[3] for retrieving the component reliability [1]. Furthermore, electrical energy not supplied to consumers is possible to compute. Probability computation of not supplied energy is only possible on the basis of reliability computation results.

A statistical significance of an outage database depends on the number of records in the database. A larger database would describe the real condition of network equipment more accurately. Therefore, it is necessary to merge databases of various distributors. The main problem of the merging is the heterogeneity feature: databases of various distributors differ from one another. In [13], we introduced a framework for retrieving reliability parameters in distribution networks. In this paper, we follow this idea, and improve the approach by new embedded database system.

This paper is organized as follows. In Section 2, we describe the reliability computation and outline the necessity of unified outage databases and we introduce characteristics of outage databases. In section 3, we present common approaches based on relational DBMS. In section 4, we introduce a new embedded DBMS called QuickDB. In Section 5, we put forward results of this system. In the last section, the paper content is resumed and the possibility of a future work is outlined.

2 Reliability Calculations

The majority of reliability computations is performed in the following way. The reliability computation of the whole system is executed on the basis of components reliability that are included in the system [9]. That is the reason why the reliability is computed in two phases. The first phase represents the retrieving of component reliability parameters and the second phase is the reliability computation itself. Other phases may include the evaluation of computed results and an improvement of the supply quality.

[2] `http://www.canelect.ca/`

[3] We have used the term 'outage database' instead of the prefered phrase 'database of failures and outages in the transmission and distribution of electrical energy' in this paper.

In virtue of experience, it is necessary to state that in most cases, the retrieving reliability parameter is far more complicated and involved than the reliability computation itself.

2.1 Failure Data As Heterogeneous Data

In the case of electrical power networks, each distributor produces incompatible outage data. Although a data model of this data may be the same (e.g. *relational data model*) [8], such data is not necessarily compatible. For example, sets of relations for two distributors belong to different relation schemes. Moreover, each scheme includes different attributes expressing the same feature of an entity type.

A common way of addressing the problem is to develop a common relation scheme and different data to transform into the relation. Another way is an application of XML technology [6]. Since we need to manage the data of a relational data model, it is necessary to transform an XML document generated by XSLT [7] to a relation. Obviously, XML is insufficient for our purpose. Yet another way is to impose this relation scheme on distributors. This solution involves the movement of the problem to another place.

Many works on data and schema integration have been published [2, 16, 18, 17]. These approaches apply various techniques, e.g. mediated schema – the schema for mapping schemes of various relational sources, wrappers, materialized views and so on for the schema integration. These works do not consider some domain-depended anomalies, e.g. the different term anomaly depicted below. Moreover, their goal is data integration, not data transformation mentioned in this paper.

2.2 Outage Database

In [13, 12], we have introduced a framework for storage and querying outage data [10, 9]. Databases of various distributors are transformed into the common relation scheme with 31 attributes (see Table 1). We see that some attributes are foreign keys of codebooks: these codebooks are labeled with an order number. Codebooks are often produced by an energy regulatory office, e.g. ERU[4] in the case of the Czech Republic. Data is stored in a storage and queried by a query language like SQL. A DBMS can be then utilized to manage this outage database.

3 Outage Data Management

Individual REAS gets the data different formatted in various data files. 31 attributes were chosen from amount attributes for the data analysis. These attributes were depicted in Table 1. Today collection contains more than 300 thousand records on voltage levels 110 kV, MV and partially LV.

[4] `http://www.eru.cz/`

Table 1. The `Outage` Relation Scheme

Order	Attribute	Data Type	Foreign Key/ Codebook Order
1	distributor	NUMBER	yes/01
2	event_order	CHAR	–
3	event_type	NUMBER	yes/02
4	distribution_point	NUMBER	yes/03
5	area	CHAR	–
6	network_type	NUMBER	yes/05
7	network_voltage	NUMBER	yes/04
8	equipment_voltage	NUMBER	yes/04
9	original_event_order	CHAR	–
10	event_cause	NUMBER	yes/06
11	equipment_type	NUMBER	yes/07
12	damaged_equipment	NUMBER	yes/08
13	damaged equipment_type	NUMBER	yes/10
14	amount	NUMBER	–
15	short_type	NUMBER	yes/09
16	producer	NUMBER	yes/11
17	production_date	DATE	–
18	T0	DATE	–
19	T1	DATE	–
20	T2	DATE	–
21	T3	DATE	–
22	T4	DATE	–
23	TZ	DATE	–
24	P1	NUMBER	–
25	P2	NUMBER	–
26	D1	NUMBER	–
27	D2	NUMBER	–
28	Z1	NUMBER	–
29	Z2	NUMBER	–
30	LxT	NUMBER	–
31	failure_type	NUMBER	yes/13

The most of today's DBMS have the ability to store such a collection and to process queries over it. A plenty of database systems have been presented during last decades. The differences between them are mainly in their architecture and data structures used. In general, we have two major database architectures: Client-Server and Embedded. We can also classify the systems as in-memory(diskless) and on-disk database systems.

3.1 Client-Server vs. Embedded DBMS

Many modern database systems use a client-server architecture, in which requests by one application (called the client) are sent to another application/service (called the server) for execution. Relational DBMS generally use the SQL language for representing requests from the client to the server. The database server then sends the answer, in the form of a table or relation, back to the client. The communication between client and server can get very complex, especially in case of large answers. Consequently, the communication time is a significant part of the whole query processing time.

An embedded database system is a DBMS which is tightly integrated with an user application. The application has direct access to stored data and therefore requires little or no ongoing maintenance.

The major representatives of the client-server architecture are the following DMBS: Oracle[5], Microsoft SQL Server[6], MySQL[7], PostgreSQL[8], Firebird/Interbase[9] and others. The major representatives of the embedded architecture are the following: Microsoft Access[10], SQLite[11], Berkeley DB[12], eXtremeDB[13] and others.

3.2 In-Memory(Diskless) vs. On-Disk Databases

An in-memory database (also known as IMDB) is a DBMS that manage data entirely in the main memory. This contrasts to traditional (on-disk) database systems, which are designed for data storage on a persistent media, e.g. disk. In-memory DBMS are faster than on-disk DBMS, because working with data in the main memory is much faster than writing to and reading from a disk. Because the design of in-memory DBMS is typically simpler than that of on-disk database systems, IMDSs can also impose significantly lower memory and CPU requirements.

[5] http://www.oracle.com/
[6] http://www.microsoft.com/sqlserver/en/us/default.aspx
[7] http://www.mysql.com/
[8] http://www.postgresql.org/
[9] http://www.firebirdsql.org/
[10] http://office.microsoft.com/en-us/access/
[11] http://www.sqlite.org/
[12] http://www.oracle.com/technetwork/database/berkeleydb
[13] http://www.mcobject.com/extremedbfamily.shtml

However, memory is not persisted, which means if a failure occurs on the server running the in-memory database, changes that have occurred in memory will be lost. In-memory DBMS then must provide own methods for recovering. Evidently, in-memory DBMS do not replace existing, more traditional on-disk DBMS, instead they complement them.

In-memory database systems are being used in conjunction with traditional on-disk DBMS. The advantage to this approach is flexibility: the developer can decide between in-memory objects and on-disk object according to performance, cost, persistence and other facts.

3.3 Query Processing in Outage data

During the reliability computation is necessary to do a several thousand complex queries. The queries may be written in general statement like:

```
select *,count(*),sum(T4-T0),sum(T3-T0)
from outage
where distributor= id_of_the_distributor
and equipment_voltage= voltage_of_the_equipment_with_an_outage
and event_type= type_of_the_event
and T0 between date_of_the_outage and date_of_the_outage
and failure_type between type_of_the_outage and type_of_the_outage
and equipment_type in ( list_of_the_equipment_types );
```

The result of this query includes the number of pieces of equipment and the sum of periods `T4-T0` and `T3-T0` for all matched records for specified period. Such a queries is not possible to efficiently process without indexing of the attributes [15]. Indexing is a technique for improving database performance. The many types of indicies share the common property that they reduce the need to examine every entry when running a query. In large databases, this can reduce query time/cost by orders of magnitude. Relational DBMS use single-dimensional data structure for indexing of particular attributes. In general, we can use only structures like the B-tree and the Hash table. These structures (because of single-dimensionality) can not increase the efficiency of the multidimensional queries presented above. The range query is processed by a sequence of searching in single-dimensional indices, and individual intermediate results are joined. Obviously, this processing is rather inefficient. There are some improvements for searching of multidimensional data, e.g. B-tree with compound keys. In this case, we create one compound index with keys related to each attribute. Obviously, this technique is not very general. If we create the composite index over the attributes *distributor*, *equipment_voltage*, and *event_type*, we have to all of them use in the query (or we can omit some of them, but only in the reverse order); otherwise the index will be not used. In next section, we present an embedded database system including multidimensional indices.

4 Embedded Database System for Handling Outage Data

In [13], a framework for retrieving reliability parameters in distribution networks was introduced. Since then, several works have been presented [12, 3, 5]. In this article, we have implemented a new relation data storage based on multidimensional data structures [19]. A variant of R-tree [11], well-known as R*-tree [4], has been applied for the implementation. Regardless to the relation storage implementation, data is queried with the SQL language.

4.1 R-tree and its Variants

Since 1984 when Guttman proposed his method, R-trees [11] have become the most cited and most used as reference data structure in this area. The R-tree is a height-balanced tree based on the B^+-tree with at least 50% utilization guaranteed. This data structure supports point and range queries and some forms of spatial joins as well. A general structure of the R-tree is shown in Figure 1.

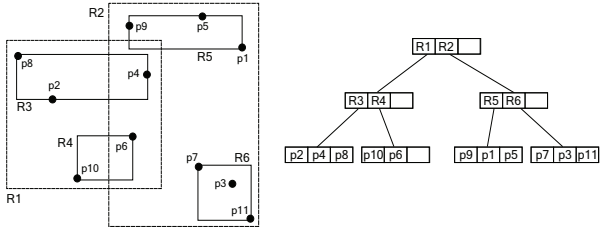

Fig. 1. Planar representation and general structure of the R-tree

It is a hierarchical data structure representing spatial data by the set of nested n-dimensional *minimum bounding rectangles* (MBR). If $\mathcal{N}$ is an inner node, it contains pairs (R_i, P_i), where P_i is a pointer to a child of the node $\mathcal{N}$. If R is the inner node MBR, then the boxes R_i corresponding to the children $\mathcal{N}_i$ of $\mathcal{N}$ are contained in R. Boxes at the same tree level may overlap. If $\mathcal{N}$ is a leaf node, it contains pairs (R_i, O_i), so called *index records*, where R_i contains a spatial object O_i. Each node of the R-tree contains between m and M entries unless it is the root and corresponds to a disk page. Other properties of the R-tree include the following:

- Whenever the number of node's children drops below m, the node is deleted and its descendants are distributed among the sibling nodes. The upper bound M depends on the size of the disk page.
- The root node has at least two entries, unless it is a leaf.

– The R-tree is height-balanced; that is, all leaf nodes are at the same level. The height of an R-tree is at most $\lfloor \log_m N \rfloor - 1$ for N index records ($N > 1$).

Many variants of the R-tree have been proposed during the last decades. Although the original R-tree algorithms tried only to minimize the area covered by MBRs, R*-tree [4] takes other objectives into account, e.g. the overlap among MBRs. R$^+$-tree [20] was introduced as a variant that avoids overlapping MBRs in intermediate nodes of the tree and an object can be stored in more than one leaf node. The Signature R-tree [14] was introduced for more efficient processing of the narrow range query; this data structure utilizes multidimensional signatures for the more efficient filtration of irrelevant tree nodes, i.e. nodes not including tuples of the query rectangle.

5 Results

In our experiments[14], we compare the presented embedded DBMS QuickDB based on the multidimensional data structure with commonly used relational database systems like Oracle, Microsoft SQL Server, Microsoft Access, Firebird/Interbase, MySQL, PostgreSQL, and SQLite. The embedded database system have been implemented in C++. The efficiency of relational DBMS has been tested in application implemented in C#.NET. The ODBC components have been used for this purpose.

The outage database includes approximately 309,000 records with 31 attributes. Efficiency of the query processing during the reliability computation was measured by processing time. We used over 200 pre-prepared sets of queries over the outage data. These queries search for records with some specific date, voltage or REAS and for concrete unit. All others attributes in the range query have intervals from zero to the domain maximal value.

Table 2. A Comparison of Query Processing Time

Database System	**Indicies [s]**		
	Without Indexing	Simple Indicies	Composite Index
Our approach	**0.467**		
Oracle	11 - 12	17 - 22	0.9 - 2
Microsoft SQL Server	18 - 21	9 - 10	16 - 17
Firebird/Interbase	25 - 33	16 - 18	1 - 1.5
MySQL	5 - 6	10 - 11	4 - 5
PostgreSQL	6 - 7	0.9 - 1.2	0.6 - 0.8
SQLite	38 - 44	25 - 27	0.5 - 0.7
Microsoft Access	53 - 54	10 - 11	56 - 58

[14] The experiments were executed on an AMD Opteron 865 1.8Ghz, 2.0 MB L2 cache; 2GB of DDR333; Windows 2008 Server.

Efficiency of the query processing in DBMS is dependent especially on capability to use simple indices or the composed index over attributes. Inappropriately chosen indices may even rapidly decrease the query processing time. In Table 2, the comparison of DBMS and our approach is presented. All queries have been processed over the data without any indices, with simple indices over the queried attributes, with one composite index over attributes distributor, voltage, and event_type. We see that QuickDB using muldimensional data structures overcomes all stated database systems. In the case of composed index, the performance of some DBMS is close to QuickDB, however composed index is not so general solution compared to the multidimensional index.

6 Conclusion

The outage database is the collection of outages in power networks in the Czech and Slovak Republics. The knowledge of them is necessary for the reliability computation of wholesale-consumer connection, therefore the demand for this computation increases. A significant number of complex queries is necessary to process during the computation. Therefore sophisticated storage of the collection is necessary. In this paper, we introduced a new embedded database system QuickDB for handling the outage database. The system is based on multidimensional data structured called R*-tree, which provides an efficient processing of complex queries over several attributes. We compared this system with common used DBMS, like Oracle, Microsoft SQL Server and several others. We showed, that our approach can process the complex queries faster up-to orders of magnitude in comparison with these DBMS.

References

1. R. E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing: Probability Models.* Holt, Rinehart and Winston, Inc., 1975.
2. C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323—-364, 1986.
3. R. Bača, M. Krátký, and V. Snášel. Bulk-loading of Compressed R-tree with Failure Data. In *Proceedings of the 4th Workshop ELNET 2007.* FEECS, VŠB–Technical University of Ostrava, 2007.
4. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings SIGMOD 1990*, pages 322–331. ACM Press, 1990.
5. P. Chovanec and M. Krátký. Benchmarking of Lossless R-tree Compression for Data of Failures in Electrical Power Networks. In *Proceedings of the 7th Workshop of ELNET, Czech Republic*, 2010.
6. W. Consortium. Extensible Markup Language (XML) 1.0, W3C Recommendation, 10 February 1998, `http://www.w3.org /TR/REC-xml`.
7. W. Consortium. XSL Transformations (XSLT), 16 November 1999, `http://www.w3.org/TR/xslt`.

8. H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems, The Complete Book*. Prentice-Hall, 2002.
9. R. Goňo and S. Rusek. Analysis of Power Outages in the Distribution Networks. In *Proceedings of the 8th International Conference on Electrical Power Quality and Utilisation (EPQU2003), Cracow, Poland*, 2003.
10. R. Goňo, S. Rusek, and M. Krátký. Reliability analysis of distribution networks. In *Proceedings of the 9th International Conference on Electrical Power Quality and Utilisation, EPQU 2007. Barcelona, Spain*. IEEE Press, 2007.
11. A. Guttman. R-Trees:ADynamic Index Structure for Spatial Searching. In *Proceedings of the International Conference on Management of Data, ACM SIGMOD 1984, Boston, USA*, pages 47–57. ACM Press, 1984.
12. M. Krátký, R. Goňo, and S. Rusek. A Framework for Querying and Indexing Electrical Failure Data. In *Proceedings of ELNET 2006. Ostrava, Czech Republic*, 2006.
13. M. Krátký, R. Goňo, S. Rusek, and J. Dvorský. A Framework for an Analysis of Failures Data in Electrical Power Networks. In *Proceedings of the International Conference on Power, Energy, and Applications Conference, ELNET/PEA 2006*. IACTA Press/IASTED, 2006.
14. M. Krátký, V. Snášel, P. Zezula, and J. Pokorný. Efficient Processing of Narrow Range Queries in the R-Tree. In *Proceedings of IDEAS 2006*. IEEE CS Press, 2006.
15. S. Lightstone, T. J. Teorey, and T. Nadeau. *Physical Database Design: The Database Professional's Guide To Exploiting Indexes, Views, Storage, And More*. Morgan Kaufmann, 2007.
16. R. Pottinger and P. Bernstein. Schema Merging and Mapping Creation for Relational Sources. In *Proceedings of the 11th International Conference on Extending Database Technology, EDBT 2008, Nantes, France*. ACM Press, 2008.
17. R. Pottinger and P. Bernstein. Schema Merging and Mapping Creation for Relational Sources. In *Proceedings of the 29th International Conference on Very Large Data Bases, VLDB 2003*, page 862–873. VLDB Endowment, 2008.
18. C. Quix, D. Kensche, and X. Li. Generic Schema Merging. In *Proceedings of the International Conference on Advanced Information Systems Engineering, CAiSE 2007*, volume 4495/2007. Springer-Verlag, LNCS, 2007.
19. H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.
20. T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R^+-Tree: A Dynamic Index For Multi-Dimensional Objects. In *Proceedings of VLDB 1997*, pages 507–518. Morgan Kaufmann, 1997.
21. The Institute of Electrical and Electronics Engineers. Guide for electric distribution reliability indices, 2003.