

# Benchmarking the Dependability of Different OLTP Systems

Marco Vieira

Polytechnic Institute of Coimbra, DEIS-CISUC  
3031 Coimbra - Portugal  
[mvieira@isec.pt](mailto:mvieira@isec.pt)

Henrique Madeira

University of Coimbra, DEI-CISUC  
3030 Coimbra - Portugal  
[henrique@dei.uc.pt](mailto:henrique@dei.uc.pt)

## Abstract

*On-Line Transaction Processing (OLTP) systems constitute the kernel of the information systems used today to support the daily operations of most organizations. Although these systems comprise the best examples of complex business-critical systems, no practical way has been proposed so far to characterize the impact of faults in such systems or to compare alternative solutions concerning dependability features. This paper presents a practical example of benchmarking key dependability features of four different transactional systems using a first proposal of dependability benchmark for OLTP application environments. This dependability benchmark is an extension to the TPC-C standard performance benchmark, and specifies the measures and all the steps required to evaluate both the performance and dependability features of OLTP systems. Two different versions of the Oracle transactional engine running over two different operating systems were evaluated and compared. The results show that dependability benchmarking can be successfully applied to OLTP application environments.*

## 1. Introduction

On-Line Transaction Processing (OLTP) systems constitute the kernel of the information systems used to support the daily operations of most of the businesses and comprise the best examples of business-critical applications. However, in spite of the importance of having dependability benchmarks for this important class of systems, the reality is that no benchmark has been proposed so far to characterize the impact of faults in such systems or to compare alternative solutions concerning dependability features.

Transactional systems industry holds a reputed infrastructure for performance evaluation and the Transaction Processing Performance Council (TPC) Benchmark™ C (TPC-C) [1] is one of the most important and well-established benchmarks. Although TPC-C specifies that dependability features of the system must ensure that data can be recovered from any point during the benchmark run, the specification does not include any procedure to evaluate the effectiveness of those dependability features or to measure their impact on the performance.

This paper presents a practical example of benchmarking key dependability features of four different transac-

tional systems using a first proposal of a dependability benchmark for OLTP application environments – the DBench-OLTP benchmark. This dependability benchmark is an extension to the TPC-C benchmark, and specifies the measures and all the steps required to evaluate both the performance and dependability features of OLTP systems.

The TPC-C standard benchmark includes two major components: a workload and a set of performance measures. The DBench-OLTP dependability benchmark adds two new elements: 1) **measures related to dependability**; and 2) a **faultload**. The faultload represents a set of faults that emulate real faults experienced by OLTP systems in the field. The measures characterize the dependability features of the system under benchmark in the presence of the faultload.

The complete specification of DBench-OLTP is available in [2]. This specification is in a form of addenda to TPC-C specification, and includes a set of extra clauses that define the new elements and some small changes needed in the benchmarking setup.

The goal of the benchmarking experiments presented in this paper is to compare and rank four different transactional systems. These systems can be considered as possible alternatives for small and medium size OLTP applications such as typical client-server database applications or e-commerce applications. In this sense, the benchmarking experiments presented give the answer to the following question: which one of the four benchmarked systems is the best choice for a typical OLTP application, considering both performance and dependability aspects?

Since the DBench-OLTP follows the benchmarking style of TPC (i.e., it is a specification to be implemented instead of a set of programs ready to run), running DBench-OLTP starting from the benchmark specification includes the following steps: 1) implementation (workload and faultload) and setup preparation for each target system; 2) running the benchmark in each system; 3) collecting the results; and 4) calculating the measures. As a final step the measures are used to compare the systems benchmarked and rank them.

The paper is organized as follows: section 2 presents an outline of the DBench-OLTP benchmark and section 3 introduces the goal of the experiments. The results are presented and discussed in section 4. Section 5 discusses the effort required to run the benchmark and section 6 concludes the paper.

## 2. DBench-OLTP specification outline

This section presents an overview of the DBench-OLTP dependability benchmark (the complete specification can be found at [2]). Since there is already an established performance benchmark for OLTP systems (the TPC-C), the DBench-OLTP dependability benchmark uses the setup, the workload, and the performance measures specified in TPC-C and adds two new elements: 1) **measures related to dependability**; and 2) a **faultload**. The faultload represents a set of faults that emulate real faults experienced by OLTP systems in the field. The measures characterize the dependability features of the system under benchmark in the presence of the faultload.

As the TPC-C consists of a detailed specification (i.e., the benchmark is a document), the new DBench-OLTP dependability benchmark extends the TPC-C specification (using an addenda) in order to define all the new elements. In order to run the DBench-OLTP dependability benchmark it is necessary to implement the TPC-C in the target system, according to the functional description provided by TPC-C specification, and the new benchmark elements (measures related to dependability and faultload) required by this new dependability benchmark. In practice, existing code and examples are adapted to new target systems, which greatly simplifies the benchmark implementation.

Figure 1 presents the test configuration required to run this dependability benchmark. As in TPC-C, the main elements are the System Under Test (SUT) and the Driver System. The SUT consists of one or more processing units used to run the workload (set of transactions submitted), and whose performance and dependability will be evaluated. The goal of the driver system is to control all the aspects concerning the benchmark run, during which it submits the workload, injects the faultload, and collects information on the SUT behavior.

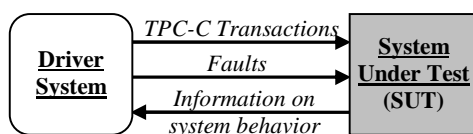


Figure 1. DBench-OLTP test configuration.

A DBench-OLTP run includes two main phases. During Phase 1, the TPC-C workload is run without any (artificial) faults. This phase corresponds to a TPC-C measurement interval, and follows the requirements specified in the TPC-C standard specification (see [1]). The goal of this first phase is to collect baseline performance measures. During Phase 2, the TPC-C workload is run in the presence of the faultload to measure the impact of faults on specific aspects of the target system dependability. As shown in Figure 2, Phase 2 is composed by several independent injection slots. An injection slot is a measurement interval during which the TPC-C workload is run and one fault from the faultload is injected.

The SUT state is explicitly restored in the beginning of each injection slot and the effects of the faults do not accumulate across different slots. The test in each injection slot is conducted in a steady state condition that represents the state in which the system is able to maintain its maximum transaction processing throughput. The system achieves a steady state condition after a given time executing transactions (steady state time).

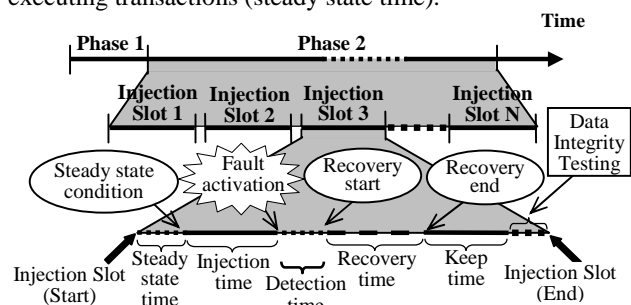


Figure 2. Benchmark run and injection slots.

A fault from the faultload is injected a certain amount of time (injection time) after the steady state condition has been achieved. Due to the fact that for some types of faults the time needed to detect the effects of a fault is highly human dependent, a typical detection time is considered for each fault. After that detection time an error detection procedure is executed to evaluate the effects of the fault and the required recovery procedure is started (if an error is detected). The recovery time represents the time needed to execute the recovery procedure. When the recovery procedure completes, the workload continues to run during a keep time to evaluate the system behavior.

After the workload end, a set of application consistency tests is performed to check possible data integrity violations caused by the fault injected. The integrity tests are performed on the application data (i.e., the data in the database tables after running the workload) and use both business rules (defined in the TPC-C specification) and the database metadata to assure a comprehensive test.

The duration of each injection slot depends on the fault injected and correspondent times (steady state time, injection time, detection time, recovery time, and keep time). However, the workload must run for at least 15 minutes after the steady state condition has been achieved.

### 2.1. Measures

The DBench-OLTP dependability benchmark is composed by three sets of measures: baseline performance measures, performance measures in the presence of the faultload, and dependability measures.

The **baseline performance measures** reported are the number of transactions executed per minute (**tpmC**) and price-per-transaction (**\$/tpmC**). These measures are inherited from the TPC-C standard benchmark and are

obtained during Phase 1. However, in the context of this dependability benchmark, these measures represent a baseline performance instead of optimized pure performance (as is the case of TPC-C), and should consider a good compromise between performance and dependability.

The **performance measures in the presence of the faultload** are:

- **Tf** – Number of transactions executed per minute in the presence of the faultload during Phase 2 (measures the impact of faults in the performance and favors systems with better fault tolerance capabilities and fast recovery).
- **\$/Tf** – Price-per-transaction in the presence of faults specified in the faultload during Phase 2 (measures the relative benefit of including fault handling mechanisms in the target systems in terms of the price).
- **Tf/tpmC** – Performance decreasing ratio due to faults.

The **dependability measures** reported are:

- **Ne** – Number of data errors detected by the consistency tests and metadata tests (measures the impact of faults on the data integrity).
- **AvtS** – Availability from the SUT point-of-view in the presence of the faultload during Phase 2 (measures the amount of time the system is available from the SUT point-of-view). The system is available when it is able to respond to at least one terminal within the minimum response time defined for each type of transaction by the TPC-C specification.
- **AvtC** – Availability from the end-users (terminals) point-of-view in the presence of the faultload during Phase 2 (measures the time the system is available from the client's point-of-view). The system is available for one terminal if it responds to a submitted transaction within the minimum response time defined for that type of transaction by the TPC-C specification. The system is unavailable for that terminal if there is no response within that time or if an error is returned.

It is worth noting that in the context of the DBench-OLTP benchmark, availability is defined based on the service provided by the system. This way, the system is considered available when it is able to provide the service defined by the transactions. For example, from the client's point-of-view the system is not available if it submits a transaction and gets no answer within the specified time or gets an error. In this case, the unavailability period is counted from the moment when a given client submits a transaction that fails until the moment when it submits a transaction that succeeds. From the server point-of-view, the system is available when it is able to execute transactions submitted by the clients. AvtS and AvtC are given as a ratio between the amount of time the system is available and the Phase 2 duration.

## 2.2. Faultload

A faultload can be based on three major types of faults: operator faults, software faults, and hardware faults. The faultload considered in this dependability benchmark is based on operator faults, as these faults are unanimously considered as one of the major causes for failures in transactional systems [3, 4].

As today's transactional systems use a database management system (DBMS) as transactional engine, transactional systems are strongly influenced by database technology, which turns the problem of operator faults in this type of system essentially into a problem of database administrator mistakes. End-user errors are not considered as operator faults, as the end-user actions do not affect directly the dependability of a DBMS.

The types of faults considered for the faultload are presented in table 1. The faultload is composed by a number of faults from these types, injected in different instants. Note that the faultload is dependent on the size and configuration of the SUT. For detailed guidelines on how to implement the faultload see [2].

Type of Fault	Target	Detection Time
Abrupt OS shutdown	Ten faults injected at the following injection times: 3, 5, 7, 9, 10, 11, 12, 13, 14, and 15 minutes.	0 Seconds
Abrupt transactional engine shutdown	Ten faults injected at the following injection times: 3, 5, 7, 9, 10, 11, 12, 13, 14, and 15 minutes.	30 Seconds
Kill set of user sessions	Five faults injected at the following injection times: 3, 7, 10, 13, and 15 minutes. The set of sessions to be killed in each fault injection must be randomly selected during the benchmark run and consists of 50% of all the active sessions from the users holding the TPC-C tables.	0 Seconds
Delete table	Three faults for each one of the following TPC-C tables: ware, order, new-order, and order-line (a total of 12 faults). The injection times to be considered are the following: 3, 10, and 15 minutes.	2 Minutes
Delete user schema	Three faults using the following injection times: 3, 10, and 15 minutes. The user to be considered is the one that holds the TPC-C tables. If the objects are distributed among several users then the user holding the greater number of TPC-C tables must be selected.	1 Minute
Delete file from disk	For each TPC-C table, the set of faults to inject is defined performing the following steps: 1) Select randomly 10% of the files containing data from the table being considered (a minimum of 1). 2) Inject 3 faults for each disk file selected, using the following injection times: 3, 10, and 15 minutes.	4 Minutes
Delete set of files from disk	Three faults for each set of files containing each TPC-C table (a total of 27 faults). The injection times are the following: 3, 10, and 15 minutes.	2 Minutes
Delete all files from one disk	The set of faults to inject is defined performing the following steps: 1) Select randomly 10% of the disks containing data from any TPC-C table (in a minimum of 1). 2) Inject 3 faults for each disk selected before, using the following injection times: 3, 10, and 15 minutes.	1 Minute

**Table 1. Faultload definition guidelines.**

### 2.3. Dependability benchmark properties

A useful benchmark must be repeatable, portable and representative [5, 6]. To be credible a dependability benchmark must report similar results when run more than once in the same environment. Concerning the DBench-OLTP benchmark, several tests have been done to confirm the fulfillment of this important requirement. It is worth noting, however, that repeatability has to be understood in statistical terms, as it is virtually impossible to reproduce exactly the same conditions concerning target system state during the benchmark run. In practice, small deviations in the measures in successive runs are normal and just reflect the asynchronous nature of transaction submission in an OLTP system. This is well known for TPC-C, as the number of transactions measured in successive runs in the same system normally has small fluctuations.

Another important property is portability, as a benchmark must allow the comparison of different systems in a given application domain. Concerning dependability benchmarking, the faultload is the component that has more influence on portability. The DBench-OLTP benchmark is portable because the faultload was defined based on the common functionalities found in all transactional systems [7] and not based on implementation details.

In order to report relevant results, a benchmark must represent real world scenarios in a realistic way. In dependability benchmarking, representativeness is mainly influenced by the workload and faultload characteristics. As we have adopted the TPC-C workload, which is well accepted as a realistic OLTP scenario, the representativeness of the workload is assured. In terms of the faultload, the type of faults used (operator faults) is unanimously considered as one of the major causes for failures in transactional systems. Furthermore, these types of faults also emulate the high-level hardware failures (e.g., disks, network, etc.) normally found in OLTP environments.

### 3. Systems under benchmarking

The goal of the benchmarking process presented in this work is to compare and rank four different transactional systems (see table 2). All these systems can be considered as possible alternatives for small and medium size OLTP applications such as typical client-server database applications or e-commerce applications (although the benchmark does not include security aspects normally needed in e-commerce). In this sense, the dependability benchmarking experiments presented in the paper give the answer to the following question: which one of the four benchmarked systems is the best choice for a typical OLTP application, considering performance and dependability aspects?

In a simplified approach, a transactional system like the ones considered in the present benchmarking experiments is based on a server that executes the transactions

submitted by several clients (client/server architecture). The server is composed by three main components: the hardware platform, the operating system, and the transactional engine. Most of the transactional systems available today use a database management system (DBMS) as transactional engine. For that reason, the DBMS is the most important component in the system. However, from the DBench-OLTP benchmark point-of-view, the SUT includes all the elements required to execute the workload and not just the DBMS.

Table 2 shows the systems that have been benchmarked. The hardware platform used is the same and the main differences among the systems are related to the operating system and transactional engine (DBMS) used. Two different versions of the Oracle DBMS and two different Microsoft operating systems have been considered.

System	OS	DBMS	Hardware Platform
2KOra8i	Win 2K Prof.	Oracle 8i (8.1.7)	• Processor: Intel PIII 800 MHz • Memory: 256MB • Disks: 4 20GB/7200rpm • Network: Fast Ethernet
XPOra8i	Win XP Prof.	Oracle 8i (8.1.7)	
2KOra9i	Win 2K Prof.	Oracle 9i (9.0.2)	
XPOra9i	Win XP Prof.	Oracle 9i (9.0.2)	

Table 2. Systems under benchmarking.

It is important to note that all systems have a similar size and have been configured to provide a good tradeoff between performance and recovery. Concerning the operating system configuration, we tried to reproduce a typical configuration of a transactional system with as many resources as possible allocated to the DBMS. The DBMS configuration has been chosen based on results from a previous work on the evaluation of the Oracle recovery mechanisms in the presence of operator faults [8].

As mentioned earlier in the paper, the faultload is dependent on the size and configuration of the system under benchmarking. It is important to note that, due to the similar size and configuration of the four SUT, the faultload used is the same for all systems. Table 3 summarizes the faultload used.

Type of fault	# of faults	% of faults
Abrupt operating system shutdown	10	10.3
Abrupt transactional engine shutdown	10	10.3
Kill set of user sessions	5	5.2
Delete table	12	12.4
Delete user schema	3	3.1
Delete file from disk	27	27.8
Delete set of files from disk	27	27.8
Delete all files from one disk	3	3.1
Total	97	100

Table 3. Faultload used in the experiments.

### 4. Benchmark results and discussion

As mentioned before, four different systems have been benchmarked. The following sub-sections present and discuss the results for each set of measures provided by the benchmark and propose a possible ranking for the SUT.

#### 4.1. Baseline performance measures

The baseline performance measures reported by the DBench-OLTP benchmark are the number of transactions executed per minute (**tpmC**) and price per transaction (**\$/tpmC**). These measures, obtained during the benchmark Phase 1, are calculated as defined in the TPC-C standard specification [1]. Figure 3 shows the baseline performance results obtained.

Results show that the baseline performance depends mainly on the DBMS used. In fact, a considerable difference in the baseline performance was observed between systems based on the two DBMS. The systems using the Oracle 9i DBMS achieve a better number of transactions executed per minute (tpmC), showing that this DBMS is quite faster than its predecessor. In terms of the price per transaction (**\$/tpmC**), the systems based on Oracle 9i present a lower cost in spite of being more expensive systems (due to the better performance reached).

#### 4.2. Performance measures in the presence of the faultload

The performance measures in the presence of the faultload reported are the number of transactions executed per minute in the presence of the faultload (**Tf**), the price per transaction in the presence of faults (**\$/Tf**), and the system performance decreasing ratio (**Tf/tpmC**). These measures characterize the impact of faults in the performance.

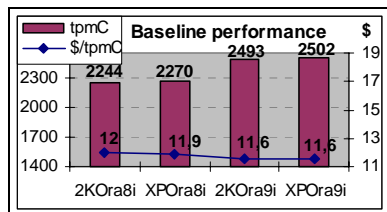


Figure 3. Baseline performance results.

Figure 4 shows the performance results in the presence of the faultload presented in Table 3. As we can see, Tf depends both on the operating system and on the DBMS used. Systems based on the Oracle 9i DBMS present better results than systems based on the Oracle 8i DBMS (similar to what happened for baseline performance), and systems running the same DBMS present different results according to the operating system used. For the systems running the Oracle8i DBMS, the Windows XP operating system is clearly more effective than Windows 2000, and for the systems running the Oracle 9i DBMS the reverse seems to occur (however, the small difference in the results does not allow a solid conclusion). However, the results show that recovery in Oracle 8i is faster when the DBMS is running over Windows XP (that is why the number of transactions in the presence of faults improves

in Oracle 8i over Windows XP), which shows that the operating system plays an important role in the recovery time in this DBMS. In Oracle 9i the influence of the operating systems on the recovery time is clearly lower, which suggests that Oracle 9i is less dependent on the operating system to process the logs needed for the recovery.

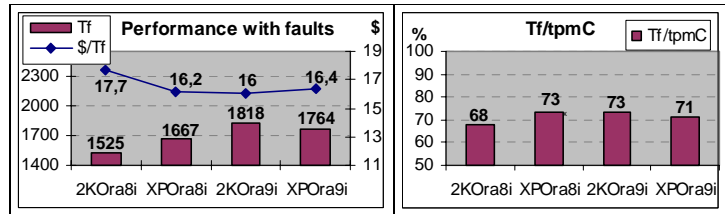


Figure 4. Performance results in the presence of faults.

Concerning the prices per transaction, the less expensive system (2KOra8i) presents the worst results (due to its poor performance in the presence of faults).

A very important aspect to observe is that the ratio between the Tf and tpmC is equal for two of the systems (XPOra8i and 2KOra9i), which shows that faults have a similar impact in a system with the Oracle 8i DBMS running over Windows XP and in a system with the Oracle 9i DBMS running over Windows 2000. For the other two systems, the impact of faults is more visible.

#### 4.3. Dependability measures

The dependability measures reported by the DBench-OLTP benchmark are the number of data integrity errors caused by faults (**Ne**), the availability from the SUT (server) point-of-view in the presence of faults (**AvtS**), and the availability from the end-users (RTE) point-of-view (**AvtC**). These measures characterize the impact of faults in the system dependability.

Figure 5 shows the availability of the systems during the benchmark run. Results show that availability from the clients point-of-view is always much lower than the availability from the server point-of-view, which seems to be normal because some types of faults affect the system in a partial way. An interesting result is the fact that the availability observed for Oracle8i DBMS over Windows XP is better than when the same DBMS is run over Windows 2000. A similar result has been observed in the systems running the Oracle 9i DBMS.

A very important conclusion is that no data integrity errors (**Ne**) were detected. This shows that the Oracle DBMS is very effective in handling faults caused by the operator.

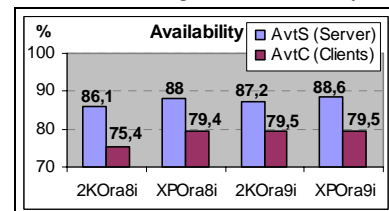


Figure 5. Availability in the presence of the faultload.

#### 4.4. Systems ranking

As mentioned before, the goal of the benchmarking presented in this work is to compare and rank four different transactional systems using the DBench-OLTP dependability benchmark. Table 4 summarizes the ranking we propose according to several criteria.

One clear conclusion is that systems running Oracle 9i are clearly better than the systems running Oracle 8i. The impact of the operating system on the measures is not so clear, as in some cases the systems using Windows 2000 are better than the systems using Windows XP and in other cases the opposite has been observed.

Concerning a global ranking, the analysis of Table 4 and all the results presented in the previous sub-sections allow us to propose the following order (from the best to the worst): 2KOra9i, XPOra9i, XPOra8i, and 2KOra8i. It is important to note that the global ranking always depends on what the benchmark performer is looking for.

Criteria	Best	2 <sup>nd</sup>	3 <sup>rd</sup>	Worst
Baseline performance	XpOra9i	2kOra9i	XpOra8i	2kOra8i
Performance with faults	2kOra9i	XpOra9i	XpOra8i	2kOra8i
Ratio Tf/tpmC	2kOra9i	XpOra8i	XpOra9i	2kOra8i
Availability	XpOra9i	XpOra8i	2kOra9i	2kOra8i

Table 4. Systems ranking.

#### 5. Benchmark execution effort

Usually benchmarking is seen as an expensive and laborious process. During the course of the present work, we had the opportunity to assess the necessary effort to implement the benchmark and to conduct the benchmarking process. Several indicators have been collected, such as: the time needed to implement TPC-C, the time needed to implement DBench-OLTP, and the time needed to conduct the benchmarking process.

In spite of being the most complex task, the implementation of the TPC-C benchmark took only about 10 days. This was possible due to the reuse of existing code and examples from several previous implementations (in a normal situation the TPC-C implementation could take more than 30 working days). The DBench-OLTP benchmark presents a similar implementation time (10 days). However, like for TPC-C we can reduce the effort needed to implement DBench-OLTP benchmark by reusing code from previous implementations (in our case this was not possible because this was the first implementation).

Concerning the time needed to conduct the benchmarking process, the time spent is very short (12 days). In fact, considering the class of systems used in this work we have been able to benchmark four different systems in about 32 days. The ratio between the total effort and number of systems benchmarked is of about 8 working days. However, it is important to note that this ratio decreases when the number of systems under benchmarking increases. Thus, we can conclude that after having the

benchmarks implemented (TPC-C and DBench-OLTP) the effort needed to benchmark additional systems is small.

#### 6. Conclusion

This paper presents a practical example of benchmarking the dependability of four different transactional systems using a first proposal of dependability benchmark for OLTP application environments – the DBench-OLTP benchmark.

Two different versions of the Oracle transactional engine running over two different operating systems have been evaluated and compared. The experimental results obtained were analyzed and discussed in detail. These results allow us to rank the four systems concerning both performance and dependability and clearly show that dependability benchmarking can be successfully applied to OLTP application environments.

The paper ends with a discussion of the effort required to run the dependability benchmark. From the indicators collected during this work, we could observe that that effort is not an obstacle for not using this kind of tools on transaction systems evaluation and comparison.

#### References

- [1] Transaction Processing Performance Consortium, “TPC Benchmark™ C, Standard Specification, Version 5.0”, 2001, available at: <http://www.tpc.org/tpcc/>.
- [2] M. Vieira and H. Madeira, “DBench – OLTP: A Dependability Benchmark for OLTP Application Environments”, Technical Report DEI-006-2002, ISSN 0873-9293, DEI – FCTUC, University of Coimbra, 2002, available at: <http://www.dei.uc.pt/~henrique/DBenchOLTP.htm>.
- [3] A. Brown and D. Patterson, “To Err is Human”, First Workshop on Evaluating and Architecting System Dependability (EASY), Joint organized with Int. Conf. on Dependable Systems and Networks, DSN-2001, Göteborg, Sweden, July, 2001.
- [4] M. Vieira and H. Madeira, “Definition of Faultloads Based on Operator Faults for DMBS Recovery Benchmarking”, 2002 Pacific Rim Intl Symp. on Dependable Computing, PRDC2002, Tsukuba, Japan, December, 2002.
- [5] J. Gray (Ed.), “The Benchmark Handbook”, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.
- [6] H. Madeira, K. Kanoun, J. Arlat, Y. Crouzet, A. Johanson and R. Lindström, “Preliminary Dependability Benchmark Framework”, DBench Project, IST 2000-25425, August, 2001.
- [7] R. Ramakrishnan, “Database Management Systems” second edition, McGraw Hill, ISBN 0 07-232206-3.
- [8] M. Vieira and H. Madeira, “Recovery and Performance Balance of a COTS DBMS in the Presence of Operator Faults”, Intl Performance and Dependability Symp. (jointly organized with DSN-2002), IPDS2002, Bethesda, Maryland, USA, June, 2002.

#### Acknowledgements

*Funding for this paper was provided, in part, by Portuguese Government/European Union through R&D Unit 326/94 (CISUC) and by DBench project, IST 2000 - 25425 DBENCH, funded by the European Union.*