First Evaluation of the Relational Implementation of ConditionsDB

C. Oliveira, A. Amorim, J. Lima, L. Pedro, N. Barros

Abstract:

The ConditionsDB is a C++ class library, built on top of a database in order to store and retrieve detector-related information with an associated "validity period".
This document is a first approach to a comparison study between the Oracle v0.4.1.6 implementation and the MySQL v0.2.6a implementation as suport databases for ConditionsDB. In order to achieve this, several local and remote performance tests were made. The comparison results are described as also the necessary steps on how to build and use both implementations.

1. Introduction

The basis of this work is to compare the Oracle and MySQL implementations of Conditions Database (ConditionsDB) as they are conceived at the present moment.
The comparison was made at a performance level by running the examples distributed with the Oracle implementation and running the same examples with the appropriate modifications on the MySQL implementation and measuring the time each on toke achieving it's purposes. New intensive usage examples were created and applied to both implementations. These intensive usage examples are based on the original ones but extended functionalities were added, such as creating more complex foldersets and folder based structures and by storing a larger number of objects and retrieving them.
Time spent running each one of this examples was measured using the Linux function *time* and theses values are the source for the comparison results that are here expressed.
Another aspect that one might consider of some relevance is that this tests may also serve as a comparison between an open-source and a commercial database solution for the support of ConditionsDB.

2. Setting up the System

Both implementations can be obtained at standard CERN AFS locations and both bring documentation on how to correctly set up the system in order to achieve compilation without errors.
An Oracle and an MySQL server must be available to support the storage and also to make available all the necessary libraries in order to correctly build the implementations.

- **MySQL**

The MySQL source code can be obtained at http://www.mysql.com as well as good documentation on how to install and use MySQL. MySQL max should also be installed to extend the MySQL functionalities. The script *mysql_install_db* must be runned in order to create all the necessary structure to start MySQL. To protect the MySQL *root* user with a password, perform:

user@machine>mysqladmin -u root password <password>

To create a user, perform:

user@machine>mysql -u root -p mysql
Enter password:

*mysql>grant all privileges on *.* to <username>@localhost identified by '<password>' with grant option;*
*mysql>grant all privileges on *.* to <username>@"%" identified by '<password>' with grant option;*

This will create a very privileged user who can connect locally and from another remote machines.

To enter MySQL perform:

user@machine>mysql -u <user> -p <password>

To list all databases, inside MySQL perform:

mysql>show databases;

To enter in a database perform:

mysql>use <database name>;

To navigate throw the database structure use SQL commands like *select * from <table name>;*

To completely erase a database, perform:

mysql>drop database <database name>;

Only notice that to access the MySQL server remotely it's necessary to add the server host to the input string:

user@machine>mysql -u <user> -p <password> -h <hostname>

Connecting the ConditionsDB implementation code to the server:

On the implementation code, the init() method is used in order to establish a connection to the MySQL server. The input string is the form:

condDBmgr->init("<host>:<database_name>:<user>:<password>");

The *database_name* input string is a general name to our choice for the ConditionsDB structure. The *host* should be *localhost* for local connections and the server host name for remote connections.

- **ORACLE**

Local Server:

For the Oracle purpose, Oracle 9i, Release 2 was used. Before Release 2 was available, Release 1 was used. Due to some interaction problems with Suse Linux 8.0 the Oracle 9i R1 installation is not trivial althought it can be used has support to ConditionsDB once the installation is acomplished sucessfully.
Oracle 9i R2 has fixed all the problems and was the one used as database server for the purpose of the tests here described. The installation under Suse 8.0 ocurred with no problem. Just a package released by Suse support on Oracle (orarun9i.rpm) was installed in order to set up the correct kernel parameters and several environment variables that are used.
Report to Appendix A to get a detailed explanation on how to have the Oracle 9i R1 – Suse 8.0 problems fixed as well on how to install the orarun9i.rpm package.
During installation, several options must be made. In the *Available Products* window *Oracle 9i Database 9.2.0.1.0* option was chosen. On the *Installation Type* window, *Enterprise Edition* was chosen. On the *Database Configuration* window, *General Purpose* option was chosen. This options already enable the Partitioning feature that is required for the process of creating the support structure for Conditions DB.

To start using a database, the database administrator must perform this some steps. It's necessary to start up and mount the database that will be used. Perform:

user@machine>export ORACLE_SID=<database name that will be used>
user@machine>oraenv
ORACLE_SID = [database name] ? (press Enter or insert database name again)
user@machine>sqlplus /nolog
*SQL*Plus>connect system as sysdba*
password:
*SQL*Plus>startup*

For shuting down a database, do *shutdown* instead of *startup.*

In order to connect the ConditionsDB implementation to Oracle, a user must be given. This user must have certain privileges. To create a user perform:

user@machine>sqlplus /nolog
*SQL*Plus>connect system as sysdba*
*SQL*Plus>password:*
*SQL*Plus>create user <username> identified by <password>;*
*SQL*Plus>grant connect to <username>*
*SQL*Plus>grant ALL PRIVILEGES to <username>*

*SQL*Plus>grant CREATE MATERIALIZED VIEW to <username>;*

To have access and navigate throw the all the structure created by ConditionsDB, a SQL*Plus console can be used. Perform:

user@machine>sqlplus <user>/<password>@<database name>

From SQP*Plus console it's possible to navigate through all the tables issuing SQL commands.
Dropping the ConditionsDB it's not at the moment a code feature. It must be performed using an external SQL script that uses all the necessary SQL commands to automatically erase all the structure. This script is named *dropCondDB.sql* comes with the Oracle implementation under the path *implementationOracle/sql.* To launch it, perform:

user@machine>sqlplus <username>/<password>@<database name>
*SQL*Plus>start /<path>/dropCondDB.sql*

The script ask for two values. The first one is the *username* and the second is the ConditionsDB database name that was passed on the init() method. This operation is irreversible and permanently deletes all the structure and data stored in the ConditionsDB database chosed.
To see the ConditionsDB database names that are already created, perform:

*SQL*Plus>select * from condition_dbs;*

Another way of dropping the ConditionsDB database is to change the value of the *STATUS* field for the chosen database from 0 to 1 on the *condition_dbs* table. This is not the same has erasing the database with the *dropCondDB* script because this action does not drop the structure, only makes the implementation rewrite the structure with what is left. For changing the value, perform;

*SQL*Plus>update condition_dbs set status=1 where status=0;*

Remote connections:

To prepare the server in order to accept connections from clients, it's necessary to start a *listener*. This is a process that resides on the database server and whose function is to listen for incoming client connection requests and manage the traffic to the server. To configure a *listener* the *Oracle Net Manager* tool may be used. Launch the tool by issuing *netmgr* on a e.g xterm. Choose *Oracle Net Configuration -> Local -> Listeners* and then *Edit -> Create*. Give the *listener* a name. Press *Add Address* and in the *Listening Locations* verify if the default values for *Protocol*, *Host* and *Port* are correct.

Under *Database Services* press *Add Database* and then modify the values for *Global Database Name, Oracle Home Directory* and *SID*. This values will identify the databases available for the remote connections and the *Global Database Name* must be passed on the SQL*Plus connect command in the client as it was defined. To start the *listener* perform:

oracle@machine>lsnrctl start

For stopping, do *stop* instead of *start*.

Oracle must also be installed in PC under the same Local Area Network (LAN) to be used as a client. All the actions listed for local server are also valid for the client. Even though it's a client installation, on the *Available Products* window, the *Oracle 9i Database 9.2.0.1.0* option must be chosen and not *Oracle 9i Client 9.2.0.1.0* because this option does not install the all Oracle header files that are necessary for the implementation to compile. In *Database Configuration* window, *Software Only* option may be chosen since no database is needed on the client. The *listener* doesn't need to be configured and started since the client don't receive connections for services but a *Net Service Name* does. This is necessary in order for the client to identify the Oracle service to access on the server. To establish this configuration, the *Oracle Net Configuration Assistant* tool may be used. Launch the tool by issuing *netca* on a e.g xterm. On the first window choose *Local Net Service Name Configuration*. Next window choose *Add a Net Service Name*. Next Window choose for which Oracle version it should be. On the next window provide the service name you want to access. Normally it should be the *Global Database Name* for the database that it will be accessed in the form *database-name.domain*. Next choose the appropriate protocol and finally the hostname of the server and if the standard port 1521 should be used or another. Give the *Net Service Name* a name and finish the Assistant. This tool writes a *tnsnames.ora* simple text file under the path stored on *$TNS_ADMIN* variable (usually */opt/oracle/product/901/network/admin* if the default options were taken). This file stores all the information given and again can alternatively be modified to our convenience instead of using the *Oracle Net Configuration Assistant* tool.
The server can then be accessed using SQL*Plus:

user@machine>sqlplus <username>/<password>@<Global Database Name>

Connecting the ConditionsDB implementation code to the server:

On the implementation code, the init() method is used in order to establish a connection to the Oracle server. The input string is the form:

condDBmgr->init("user=<username>,passwd=<passwd>,db=<Oracle_database_name>,cond_db=<conditionsdb_database_name");

The *db* input string should the Oracle database name (specified in *$ORACLE_SID* variable) for local connection and the *Global Database Name* for the remote connection. *cond_db* input string is a general name to our choice for the ConditionsDB structure.

3. Running Examples

All the tests were performed in a PC with the following configuration:

- Hardware:
  - Intel Pentium 4, 1,6GHZ
  - 756Mb DDR RAM PC133MHz
  - 30 Gb HDD, ATA100, 7200 RPM

- Operating System:
  - Suse 8.0, linux kernel 2.4.18-64GB-SMP

- Database Servers:
  - Oracle: Oracle9i Enterprise Edition Release 9.2.0.1.0 – Production
    With the Partitioning, OLAP and Oracle Data Mining options
    JServer Release 9.2.0.1.0 - Production
  - MySQL: Ver 11.15 Distrib 3.23.48

Bellow is the list of all the examples used on the tests, as well as a brief description of what each one performs. All this examples are distributed in the 0.4.1.6 distribution of the Oracle implementation and were migrated to the MySQL implementation. This migration is easily accomplished with some minor changes in the code.
From the common code:

```
#include <ConditionsDB/CondDBXXXXMgrFactory.h>
        ...
ICondDBMgr* CondDBmgr = CondDBXXXX MgrFactory::createCondDBMgr();
        ...
        CondDBmgr->init();
        ...
CondDBXXXXMgrFactory::destroyCondDBMgr( CondDBmgr );
```

XXXX must be changed for the specific implementation. For MySQL must be changed to MySQL and for Oracle to OracleDB.

Tests:

- *createFolders* – verifys if a certain structure exists. Should not exist, creates it and establish a connection. The structure defined is the folder *temp* under the folderset *cal*.
- *storeData* and *exampleObject* are object store procedures under a folder. Should not exist this folder, it creates it.
- *readData* and *exampleObjectRead* are object reading procedures of the previously stored objects.
- *comprehensiveTest* is an extended funcionality of all the procedures (NOTE - this test was not migrated to MySQL).

To measure each ones performance, the *time* Linux function was used. Syntax is like:

user@machine:(...)/tst>time ./<example_name>

From the manual entries for function *time* comes:

time - time a simple command or give resource usage
Returns three arguments to the standard output:
real - the elapsed real time between invocation and termination
user - the user CPU time
sys - the system CPU time

Bellow is listed the *real* argument output for each test. Report to Appendix B for detailed test results with all arguments.

- **MySQL results:**

	MySQL (local)	***MySQL (remote)***
createFolders	real 0m0.134s	real 0m0.133s
storeData	real 0m0.065s	real 0m0.065s
readData	real 0m0.023s	real 0m0.046s
exampleObject	real 0m0.038s	real 0m0.060s
exampleObjectRead	real 0m0.021s	real 0m0.053s

Table 1
time arguments for the MySQL implementation of ConditionsDB (local and remote)

- **Oracle results:**

	Oracle (local)	***Oracle (remote)***
CreateFolders	real 0m11.101s	real 0m23.086s
storeData	real 0m0.427s	real 0m0.633s
readData	real 0m0.130s	real 0m0.228s
exampleObject	real 0m0.233s	real 0m0.293s
exampleObjectRead	real 0m0.131s	real 0m0.248s
comprehensiveTest	real 6m55.043s	real 7m42.640s

Table 2
time arguments for the Oracle implementation of ConditionsDB (local and remote)

4. Intensive Usage

The following tests were implemented in order to test the implementation in it's functionalities and to see it's comportment under intensive structures creation and intensive storage and reading procedures. The tests use the same common structure of the implementations and the only changes between them in order to make them run are again changing the include file, the constructor and the destructor as already explained.

- *createFolderx* – creates a folder which is meant to be under a certain folderset structure. The code creates the required foldersets before creating the folder. In this example it creates three folderset levels and then creates the folder but it can create as more as needed.
- *storeDatax* – creates a user given number of generic objects under a folder. Should not exist this folder, it creates it.
- *readDatax* – reads all the previously stored objects with a description to the standard output.

For the *storeDatax* test, several amounts of objects were stored in order to extrapolate a tendency for the relation *objects stored vs. time spent to store* and for *objects stored vs. time spent to read* .

- **MySQL results:**

Intensive Usage		***MySQL (local)***	***MySQL (remote)***
createFolderx		real 0m0.034s	real 0m0.072s
storeDatax	*10 objs*	real 0m1.447s	real 0m1.127s
	100 objs	real 0m1.368s	real 0m2.345s
	1.000 objs	real 0m4.056s	real 0m6.341s
	10.000 objs	real 0m23.175s	real 0m56.929s
	50.000 objs	real 1m40.040s	real 3m53.565s
	100.000 objs	real 3m49.184s	real 8m16.510s
readDatax	*10 objs*	real 0m0.037s	real 0m0.085s
	100 objs	real 0m0.156s	real 0m0.415s
	1.000 objs	real 0m1.429s	real 0m3.050s
	10.000 objs	real 0m36.069s	real 0m43.630s
	50.000 objs	real 6m10.393s	real 6m45.314s
	100.000 objs	real 21m23.881s	real 22m51.999s

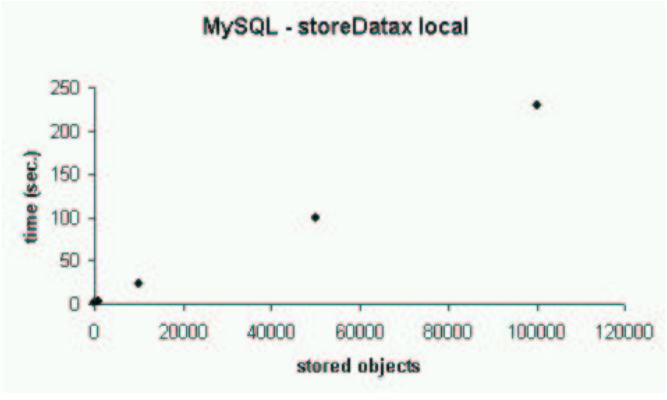
Table 3
time arguments for the MySQL implementation of ConditionsDB with intensive usage tests (local and remote)

- **Oracle results:**

Intensive Usage		***Oracle (local)***	***Oracle (remote)***
createFolderx		real 0m15.173s	real 0m11.857s
storeDatax	*10 objs*	real 0m4.973s	real 0m5.434s
	100 objs	real 0m9.749s	real 0m15.820s
	1.000 objs	real 1m2.375s	real 1m2.850s
	10.000 objs	real 9m22.103s	real 11m12.554s
	50.000 objs	real 53m48.330s	real 52m49.003s
	100.000 objs	real 109m40.878s	real 104m13.283s
readDatax	*10 objs*	real 0m0.324s	real 0m0.955s
	100 objs	real 0m1.403s	real 0m3.061s
	1.000 objs	real 0m14.033s	real 0m27.556s
	10.000 objs	real 2m1.919s	real 4m37.315s
	50.000 objs	real 11m0.185s	real 25m16.651s
	100.000 objs	real 25m46.423s	real 46m53.846s

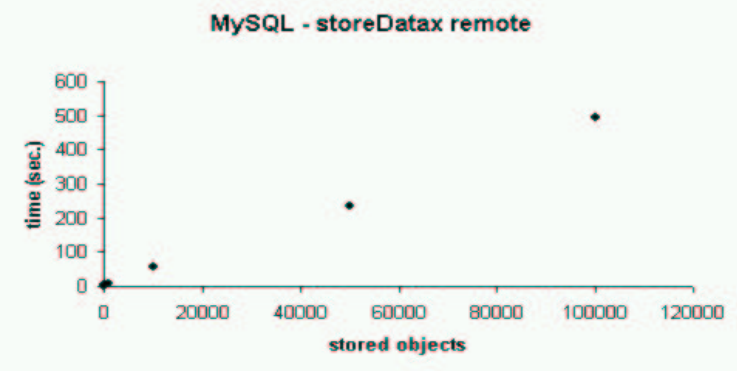
Table 4
time arguments for the Oracle implementation of ConditionsDB with intensive usage tests (local and remote)

- **Graphic results:**

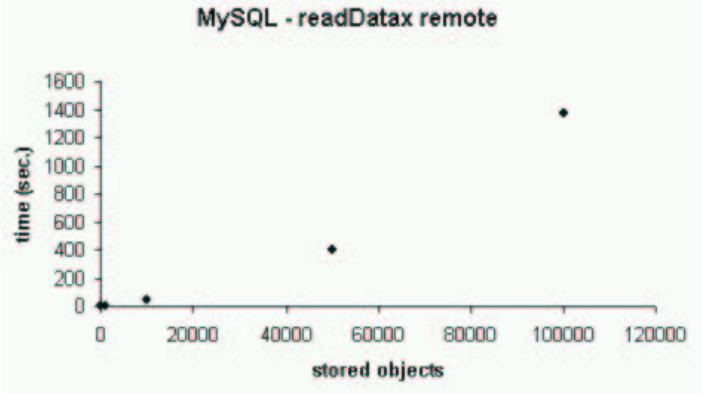


Graph diagram 1 – objects stored vs. time spent storing for MySQL implementation (local)

Graph diagram 2 – objects read vs. time spent reading for MySQL implementation (local)



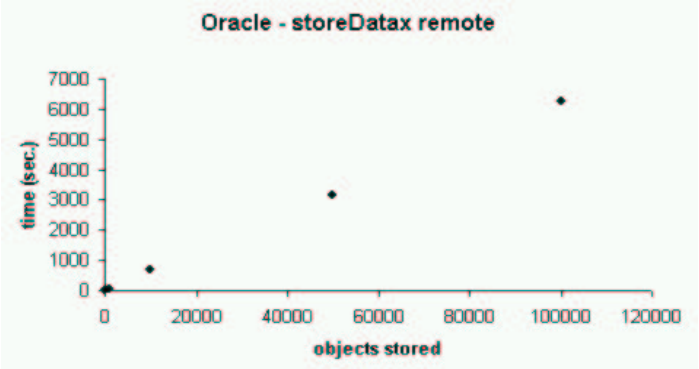
Graph diagram 3 – objects stored vs. time spent storing for MySQL implementation (remote)



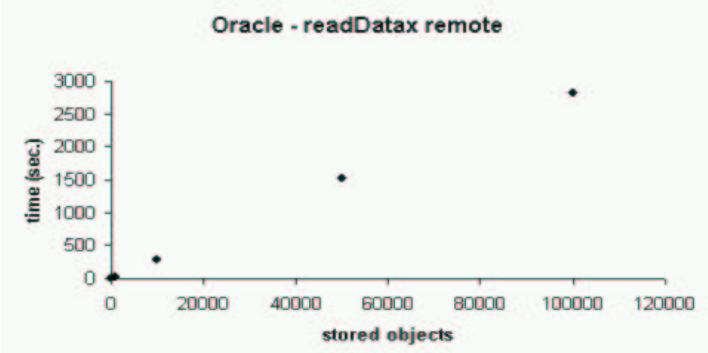
Graph diagram 4 – objects read vs. time spent reading for MySQL implementation (remote)

Graph diagram 5 – objects stored vs. time spent storing for Oracle implementation (local)

Graph diagram 6 – objects read vs. time spent reading for Oracle implementation (local)



Graph diagram 7 – objects stored vs. time spent storing for Oracle implementation (remote)



Graph diagram 8 – objects read vs. time spent reading for Oracle implementation (remote)

5. Conclusions

At the present moment as both implementations are conceived, MySQL presents itself in all tests as a quicker solution. The MySQL implementation doesn't present indexes but introducing this feature at a future version can bring even more satisfactory results reading objects but increasing the time spent on storing.
Both implementations present a growing linear relation tendency in the objects stored vs. time spent as well as in objects read vs. time spent.
More user friendly options on setting up the system for the MySQL implementation and less hardware sources requirement. Although, Oracle presented itself as a very powerful database with many assistants that help bringing the most benefit of all features.

APPENDIX A

Installation of the Oracle 9i, Release 1, database in the Suse Linux distribution, versions 7.3/8.0

Introduction

This document is intended to be an explanation of the necessary steps for the installation of the Oracle 9i. Release 1, database on the Linux operative system Suse, versions 7.3 and 8.0. The usual installation of Oracle through the Installer generates several errors when linking the necessary libraries. It is necessary to perform some tasks to successfully conclude the installation.
Release 2 of Oracle 9i solved all this problems and the installation is achieved with no problem ocurring. Still, package orarun9i.rpm developed and made available by Suse support on Oracle should be installed as described since it facilitates the installation.
The procedures were taken among the site of Oracle[1] and the support site of Suse for Oracle[2].

1 – Hardware requirements

- PC Intel or AMD with CPU 32bit
- 256 MB RAM minimum
- 4 GB of hard disk space

2 – Detailed Installation:

1. For simplification of the process, two terminals can be used. One to perform root privileged commands and another for the oracle user, explained ahead.

Due to a flaw of memory in the Oracle Installer , the process of installation needs about 1,5 GB of RAM memory (physics and swapspace). One way to solve the problem in case of not having this amount of memory is to temporarily create a swap file that can be removed after the installation.
In the terminal root, follow the steps:

• **dd if=/dev/zero of=/swapfile bs=1k count=1048576**
Creates a 1GB file (or any other value in" count")

• **mkswap /swapfile**
Turns the file on a swap file

• **swapon /swapfile**
Activates the swap file

NOTE: After concluded the installation, it is possible to remove the file. However, it is necessary deactivate it first. To do that, make **swapoff /swapfile**

2. Install the orarun9i.rpm package

```
rpm -Uvh orarun9i.rpm
```

This package is distributed by Suse and creates the necessary environment variables such as $ORACLE HOME, configures the necessary kernel parameters for Oracle and initiates / finalises the database during the startup / shutdown.

3. Verify and edit the **/etc/rc.config.d/oracle.rc.config** file.

This is an optional step, since the parameters by default are correct.

4. Verify and edit the **/etc/profile.d/oracle.sh** file.

There are three important environment variables:

- ORACLE_BASE is the base path for everything that will be installed, as Oracle products, Java Runtime Environment (JRE), that id used to run several tools, the documentation and the inventory where Oracle Installer Oracle keeps the information of the installed components. By default, is also the base for the database files, logfiles and parameter files. It is, however, possible to specify a different location to keep the databases by the time of it's creation.

- ORACLE_HOME is the path below ORACLE BASE where Oracle database is installed.

- ORACLE_SID is an identifier for a database. For each database running simultaneously in the same machine a different SID is attributed. It's suggested to use only four characters for SID, however, it is possible to use more. By default, SID is named *mydb*.

By default, Suse points ORACLE_BASE for / opt/oracle. In case nothing is altered, it will be where Oracle will be installed.

5. The installation will be made by an Oracle user. To define the password for the this user, execute the command:

passwd oracle

6. To begin the installation, it is necessary to switch for the oracle user. On the terminal destined to the oracle user, execute the command:

sux - oracle

7. It is necessary to deactivate the NumLock key. Otherwise, pressing the Installer buttons or using the mouse will sort no effect. The cause of this is related with a bug in Installer or in Java.

8. Using the files that are available in the Oracle site[1] for download follow the steps:

- Place the three files on the *home* path of oracle user

– Linux9i Disk1.cpio.gz (412,092kb)
– Linux9i Disk2.cpio.gz (638,547kb)
– Linux9i Disk3.cpio.gz (82,956kb)

- Decompress the files using **gunzip**

  – Ex: gunzip Linux9i Disk1.cpio.gz

- Extract the files using **cpio -idmv < <filename>**

  – Ex: cpio -idmv < Linux9i Disk1.cpio

9. After extracting all the files, three directories are created, *Disk1*, *Disk2* and *Disk3*. From *Disk1*, execute *runInstaller*. that will begin Oracle Universall Installer.

Oracle Universall Installer

10. The welcome window appears. Press Next.

11. On the first installation of an Oracle product it will appear the Inventory Location window. It is necessary to specify a location where Universal Oracle Installer will keep a track (inventory) of which software and components are installed. The suggested place is ORACLE_BASE. It is recommended to accept the suggestion.

12. The next two windows will appear in case the */etc/oraInst.loc* file doesn't exist which will be created at this moment. The second step identifies the user that will have permission to update the Oracle software. It is possible to indicate which user is intend to give permission to do that task. In case the space is left blank, that permission will be given to the user that began Installer, in this case the oracle user.

13. After pressing *Next*, a text box will appear indicating that it is necessary to perform certain actions that need root privileges. In the root terminal, execute the script as explained. After that, press *Continue*.
    These last two steps create the */etc/oraInst.loc* global file for the Oracle configuration that will be used by Oracle Universall Installer or by another Oracle product to know which components are already installed. In case this file already exists, these last two steps are ignored by the Installer.

14. The following step treats of the file locations. The ones inserted by default are the suitable ones. Press *Next*.

15. Choose the product for installation. For the purpose of this document, the *Oracle Database <version>* option was chosen. Press *Next.*

16. Choose the kind of installation. For the purpose of this document, the *Enterprise Edition* was chosen.

17. For the Database configuration, a *General Purpose* was chosen.

    NOTE: It might happen that in case of other option being chosen in one of the steps above no problem occurs with the installation and everything goes normal.

18. Database identification. Insert a name and an identifier.

19. Characters style for the database. This choice is important because after the database creation the style cannot be altered.

20. Specify the location of JDK (Java Development Kit) in the system.
    E.g.: / usr/lib/jdk1.1.8
    From the release notes of Oracle9i, comes:
    - **JRE**: Oracle components require 1.1.8v3 version except BC4J, UltraSearch and JDBC 1.2 drivers, which require Sun JDK/JRE 1.3.1 version.
    - Oracle HTTP server Apache requires JDK 1.3.1

21. A summary of all the installation is displayed. Press *Install*.

22. After the components are installed, Installer will proceed with the linking of the necessary libraries. At this time the problems begin originated by the Oracle installation scripts.
    As a resolution, do the following steps:

    - When the first window to indicate a mistake appears (related with plsql), open a terminal and login as oracle user.

    - Go to $ORACLE_HOME/bin

    - Edit the ***genclntsh*** script

    - Add to the end of the line

      -lgcc -L/usr/lib/gcc-lib/i486-suse-linux/2.95.3/

The all line would look like this:

SYSLIBS=‘cat ${ORACLE HOME}/lib/sysliblist‘ ” -ldl -lm -lc-lgcc -L/usr/lib/gcc-lib/i486-suse-linux/2.95.3/”

- Execute ***genclntsh*** and wait until the script report that the necessary library was created.
- Continue the Installer by pressing *Retry*.

23. Concluded the installation a window appears indicating that it is necessary to perform certain actions that need root privileges. In the root terminal, execute the suitable script just as explained in the window. Press *Ok*.

24. The Installer will proceed now to the configuration of the net services, to the creation of the database and starting up the Apache webserver that comes with Oracle.
    At this moment, Oracle is completely installed and the possible failure of one of the configuration tools doesn't have any importance, once each one of them can be executed and configured later.
    Concluded these configurations, a summary is exhibited containing the name of the database, the system identifier for the database and the passwords for DBA (Database Administrator).

25. End of installation. Click *Next* to install other components like Pro-C/C++.
    Click *Exit* to quit Installer.

References:

[1] http://www.oracle.com
[2] http://www.suse.com/en/support/oracle/db/9i 80.html

APPENDIX B

Detailed test results for the MySQL and Oracle implementation of ConditionsDB

MySQL implementation v0.2.6a: local / remote

	MySQL (local)	***MySQL (remote)***
createFolders	real 0m0.134s user 0m0.010s sys 0m0.010s	real 0m0.133s user 0m0.020s sys 0m0.000s
storeData	real 0m0.065s user 0m0.010s sys 0m0.000s	real 0m0.065s user 0m0.020s sys 0m0.000s
readData	real 0m0.023s user 0m0.020s sys 0m0.000s	real 0m0.046s user 0m0.020s sys 0m0.000s
exampleObject	real 0m0.038s user 0m0.020s sys 0m0.000s	real 0m0.060s user 0m0.020s sys 0m0.000s
exampleObjectRead	real 0m0.021s user 0m0.010s sys 0m0.010s	real 0m0.053s user 0m0.010s sys 0m0.000s

Intensive Usage	***MySQL (local)***	***MySQL (remote)***
createFolderx	real 0m0.034s user 0m0.010s sys 0m0.010s	real 0m0.072s user 0m0.020s sys 0m0.000s

Intensive Usage		***MySQL (local)***	***MySQL (remote)***
storeDatax	*10 objs*	real 0m1.447s user 0m0.010s sys 0m0.010s	real 0m1.127s user 0m0.020s sys 0m0.010s
	100 objs	real 0m1.368s user 0m0.050s sys 0m0.050s	real 0m2.345s user 0m0.030s sys 0m0.040s
	1.000 objs	real 0m4.056s user 0m0.320s sys 0m0.260s	real 0m6.341s user 0m0.170s sys 0m0.150s
	10.000 objs	real 0m23.175s user 0m2.390s sys 0m2.270s	real 0m56.929s user 0m1.910s sys 0m1.800s
	50.000 objs	real 1m40.040s user 0m12.700s sys 0m11.480s	real 3m53.565s user 0m9.150s sys 0m8.020s
	100.000 objs	real 3m49.184s user 0m26.970s sys 0m24.240s	real 8m16.510s user 0m17.880s sys 0m16.760s

Intensive Usage		***MySQL (local)***	***MySQL (remote)***
readDatax	*10 objs*	real 0m0.037s user 0m0.000s sys 0m0.010s	real 0m0.085s user 0m0.010s sys 0m0.020s
	100 objs	real 0m0.156s user 0m0.050s sys 0m0.020s	real 0m0.415s user 0m0.040s sys 0m0.010s
	1.000 objs	real 0m1.429s user 0m0.280s sys 0m0.130s	real 0m3.050s user 0m0.270s sys 0m0.090s
	10.000 objs	real 0m36.069s user 0m13.970s sys 0m2.110s	real 0m43.630s user 0m10.920s sys 0m2.840s
	50.000 objs	real 6m10.393s user 4m10.360s sys 0m12.590s	real 6m45.314s user 4m31.840s sys 0m21.280s
	100.000 objs	real 21m23.881s user 15m47.960s sys 0m26.670s	real 22m51.999s user 17m1.280s sys 0m45.680s

Oracle implementation v0.4.1.6: local / remote

	Oracle (local)	***Oracle (remote)***
createFolders	real 0m11.101s user 0m0.060s sys 0m0.020s	real 0m23.086s user 0m0.090s sys 0m0.110s
storeData	real 0m0.427s user 0m0.030s sys 0m0.010s	real 0m0.633s user 0m0.030s sys 0m0.030s
readData	real 0m0.130s user 0m0.010s sys 0m0.010s	real 0m0.228s user 0m0.050s sys 0m0.010s
exampleObject	real 0m0.233s user 0m0.030s sys 0m0.000s	real 0m0.293s user 0m0.010s sys 0m0.030s
exampleObjectRead	real 0m0.131s user 0m0.030s sys 0m0.020s	real 0m0.248s user 0m0.040s sys 0m0.030s
comprehensiveTest	real 6m55.043s user 0m11.550s sys 0m1.910s	real 7m42.640s user 0m12.640s sys 0m2.280s

Intensive Usage		***Oracle (local)***	***Oracle (remote)***
createFolderx		real 0m15.173s user 0m0.020s sys 0m0.020s	real 0m11.857s user 0m0.060s sys 0m0.040s
storeDatax	*10 objs*	real 0m4.973s user 0m0.030s sys 0m0.000s	real 0m5.434s user 0m0.060s sys 0m0.030s
	100 objs	real 0m9.749s user 0m0.140s sys 0m0.030s	real 0m15.820s user 0m0.130s sys 0m0.050s
	1.000 objs	real 1m2.375s user 0m0.830s sys 0m0.120s	real 1m2.850s user 0m1.150s sys 0m0.170s
	10.000 objs	real 9m22.103s user 0m8.470s sys 0m1.510s	real 11m12.554s user 0m10.110s sys 0m1.630s
	50.000 objs	real 53m48.330s user 0m41.140s sys 0m5.960s	real 52m49.003s user 0m48.950s sys 0m7.220s
	100.000 objs	real 109m40.878s user 1m26.100s sys 0m12.610s	real 104m13.283s user 1m35.490s sys 0m15.380s

Intensive Usage		***Oracle (local)***	***Oracle (remote)***
readDatax	*10 objs*	real 0m0.324s user 0m0.040s sys 0m0.010s	real 0m0.955s user 0m0.100s sys 0m0.010s
	100 objs	real 0m1.403s user 0m0.360s sys 0m0.080s	real 0m3.061s user 0m0.530s sys 0m0.150s
	1.000 objs	real 0m14.033s user 0m3.100s sys 0m0.650s	real 0m27.556s user 0m4.680s sys 0m0.970s
	10.000 objs	real 2m1.919s user 0m36.590s sys 0m6.860s	real 4m37.315s user 0m43.370s sys 0m7.750s
	50.000 objs	real 11m0.185s user 2m57.010s sys 0m35.400s	real 25m16.651s user 3m38.370s sys 0m43.390s
	100.000 objs	real 25m46.423s user 6m0.550s sys 1m11.110s	real 46m53.846s user 7m3.400s sys 1m18.220s