Table of Contents

? versions matrix

Current Status (*Actions Log*)

? integration service

- prepare service running
    - ♦ 4.0.xx (to be dropped soon as the support life-time has ended for this MySQL ? version)
    - ♦ **DONE!** 5.0.xx
    - ♦ 5.1.xx MySQL ? servers
    - ♦ version 4.1.xx won't be used as it has been very short time in production to be used widely (can be put in place if needed on request)
- Define database user accounts and schemas needed for all the tests to be run
    - ♦ **DONE!** COOL accounts + schemas
    - ♦ **DONE!** POOL accounts + schemas
    - ♦ **DONE!** Persistency accounts + schemas
- update and re-factoring of the existing test suite in Persistency to be integrated into LCG nightly build & test system so the whole test suite runs on periodic basis and uploads results to this web area
    - ♦ **DONE!** Persistency integration tests using MySQLAccess already succeed

Introduction

LCG AA CORAL project provides relational abstraction layer which allows to access relational data in technology agnostic way. The number of RDBMS back-ends are supported like Oracle, MySQL ?, SQLite or webcache based Frontier system. The MySQL ? back-end implementation is a bit particular w.r.t the other RDBMS back-ends due to the number of released versions being used in parallel. This includes client libraries as well as server versions and it implies one needs to understand the differences among them. One can find much of the knowledge reading the MySQL ? documentation but there is more to that when it comes to performance impact of using a particular client version against a different version of the server or switching data protocol from one to another (this is another specific feature of MySQL ?).
So far only the functional tests have been done as the integral part of the Persistency release process but not much of the performance test matrix has been in place. The reasons are the time constraints coming from the focus on the Persistency features to be put in place for production level use and the still growing list of requirements and use cases coming from the Persistency users. In the short time scale there will be a dedicated MySQL ? test integration facility installed on robust hardware connected to RAID storage and aligned with the recent migration to CMT based configuration and build management framework which uses LCG nightly build facility executing Persistency test systematically. The test matrix thus will be filled soon with more data and updated regularly.

The MySQL versions jungle

MySQL ? RDBMS software has been growing very rapidly during the last couple of years leading to a very complex situation for its users and DBAs. One can choose these days from five! different version of MySQL ? database software, namely 3.2.xx, 4.0.xx, 4.1.xx, 5.0.xx and 5.1.xx being quickly pushed into production use, reaching the final cycle of beta testing phase. With the growing major releases number the feature list is changing very much and even backward incompatible changes were/are introduced along with the new functionality.
The changes affect the data protocol between MySQL ? client and server switching from ASCII to binary one, adding networked cluster functionality too and other changes like prepared statements API and server side cursors as well as stored procedures.

The following table tries to capture the current state of the affairs:

Version	**Protocol**	**Prepared statement API**	**Server side cursors**	**Stored procedures**	**BLOB/CLOB handling**
3.2.xx	ASCII	no	no	no	tbd
4.0.xx	ASCII	no	no	no	tbd
4.1.xx	binary*	yes	no	no	tbd
5.0.xx	binary*	yes	yes***	yes	tbd
5.1.xx	binary*	yes	yes***	yes	tbd

* - this includes password hashing change ruling out the older clients unless one uses extra server settings
** - trying to use it against older server does not work
*** - read-only

LCG supported ? versions

Persistency project (formerly POOL RAL sub-project) started supporting MySQL ? relational data access via ODBC interface in order to keep semantics as close as possible to the Oracle OCI client API. This was working well till the moment MySQL ? MyODBC ? package got broken as of the 5.x.x version for quite some time, which has become a show stopper for the further Persistency MySQLAccess ? development. The migration to the native MySQL ? C API has resolved the problem of being blocked but introduced more complexity to the code due to the many versions of the MySQL ? client library with different features.

The LCG supported MySQL ? versions are:

POOL/RAL (UnixODBC ? bridge + MyODBC ? driver)

- 3.23
- 4.0.13 - 4.0.26

CORAL (4.0.xx client)

- 3.23
- 4.0.xx
- 4.1.xx (with turning to old password hashing scheme)
- 5.0.xx (with turning to old password hashing scheme) *

CORAL (5.0.xx client)

- 3.23
- 4.0.xx
- 4.1.xx
- 5.0.xx *

* - partially tested in Persistency and received more feedback from Persistency users

? 4.0.xx tests

So far done, the release time performed, testing has revealed the following constraints on the use of the MySQL ? client library version 4.0.xx:

- ASCII protocol forbidding prepared statement API
  - lack of bind variables
  - hard parsing
  - lack of bulk DML statements (insert,update, delete)
- Missing cursor concept
  - no way to tune pre-fetch size
  - direct implication is that the current Persistency uses ***store of the whole result set*** policy which may crash a client if a result set is larger than its available virtual memory can hold
  - use of fetching a single row at a time is not considered as a viable option since it kills performance due to the frequent round-trips to a server when fetching data
- Connection failure or degraded security if used against a server version 4.1.xx or newer

Apart from the last point, results apply in general to any MySQL ? server version used for testing.

? 5.0.xx tests

The functionality testing, as done so far, shows the following constraints:

- though binary protocol enables use of prepared statements and bind variables it can't be used against older servers from 3.23 and 4.0.xx series
- still lack of bulk data transfer API
- the new cursor option starting from 5.0.xx server versions removes the constraints of 4.0.xx client, however, its implementation is still troublesome for some large result sets as it affects negatively performance due to the fact that cursor is implemented as in memory temporary table but turns into a MyISAM ? table when it reaches a certain threshold.

- DECIMAL(p,n) SQL type affects current Persistency SQL type conversion from/to C/C++

To be done

Persistency development

- unify the existing functionality test infrastructure in Persistency MySQLAccess ? so it can be comfortably executed by LCG nightly build & test system
- integrate the missing performance tests
- **DONE!** built-in server version auto-detection
- finish the use of the prepared statement API in Persistency MySQLAccess ? so the performance comparisons can be performed by MySQL ? integration service
- provide customizable use of MySQL ? cursors
- **OPTIONAL**: use of the new MySQL ? 5.0.x DATA DICTIONARY for querying schema and table meta data

Test metrics

The following tests should be performed against each server version:

1. Time to fetch schema and table meta data
    - ♦ COOL does it often
2. Time to insert fixed, variable strings and CLOBs
3. Time to insert numerical data: integer, floating-point and mixed
4. Time to insert BLOBs
5. Time to bulk insert data
6. Time to query & fetch CLOBs
7. Time to query & fetch BLOBs
8. Time to query & fetch fixed and variable length strings
9. Time to query & fetch numerical data
10. Time to query & fetch mixed data
11. Time to query & fetch from dependent tables having relations (foreign keys)
12. Compare times for store result & using cursor cases (where applicable)
13. Compare times for ASCII & binary protocol using prepared statements (where applicable)

Optionally one can perform the tests using a different storage engine (Persistency defaults to InnoDb ?) to see how it affects the overall performance.

E dit | W YSIWYG | A ttach | P rintable | R aw View | Backlinks: We b, A l l Webs | H istory: r7 < r6 < r5 < r4 < r3 | M ore topic actions
PSSGroup

- **PSSGroup**

- **On Shift**

- **Sub-wikis**
- PhysicsDatabases
- DBA Area
- 3D project

- **Other**
- PSS group public
- PhyDB public
- DBA Services Wiki

- **PSSGroup Wiki**
- Last Changes
- Pages Index
- Search web

-
-
- |

Copyright by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback

This topic: PSSGroup > MySQLTesting
Topic revision: r9 - 2010-06-11 - PeterJones

Copyright &© 2008-2020 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback