```
diff -crBN moodle-1.9.6/.hg_archival.txt moodle-db2-tip/.hg_archival.txt
*** moodle-1.9.6/.hg_archival.txt   1969-12-31 16:00:00.000000000 -0800
--- moodle-db2-tip/.hg_archival.txt 2009-12-11 16:54:12.000000000 -0800
***************
*** 0 ****
--- 1,2 ----
+ repo: 44c5e7d604d31f974e3982619113080008ebbd9e
+ node: e3c050177f246d7ca419a84403b62acdfa6744fb
diff -crBN moodle-1.9.6/.hgtags moodle-db2-tip/.hgtags
*** moodle-1.9.6/.hgtags    1969-12-31 16:00:00.000000000 -0800
--- moodle-db2-tip/.hgtags  2009-12-11 16:54:12.000000000 -0800
***************
*** 0 ****
--- 1 ----
+ dbb04744115c119144b78cf8b803e55277860aa7 pmn-20091101-base
diff -crBN moodle-1.9.6/moodle/admin/environment.xml moodle-db2-
tip/moodle/admin/environment.xml
*** moodle-1.9.6/moodle/admin/environment.xml   2009-10-08 17:06:03.000000000 -0700
--- moodle-db2-tip/moodle/admin/environment.xml 2009-12-11 16:54:12.000000000 -0800
***************
*** 145,150 ****
--- 145,151 ----
        <VENDOR name="odbc_mssql" version="9.0" />
        <VENDOR name="mssql_n" version="9.0" />
        <VENDOR name="oracle" version="9.0" />
+       <VENDOR name="db2" version="9.0" />
      </DATABASE>
      <PHP version="4.3.0" level="required">
        <RESTRICT function="restrict_php50_version" message="php50restricted" />
***************
*** 221,227 ****
        <VENDOR name="mssql" version="9.0" />
        <VENDOR name="odbc_mssql" version="9.0" />
        <VENDOR name="mssql_n" version="9.0" />
!       <VENDOR name="oracle" version="10.2" />
        <VENDOR name="sqlite" version="2.0" />
      </DATABASE>
      <PHP version="5.2.8" level="required">
--- 222,228 ----
        <VENDOR name="mssql" version="9.0" />
        <VENDOR name="odbc_mssql" version="9.0" />
        <VENDOR name="mssql_n" version="9.0" />
!       <VENDOR name="oracle" version="10.0" />
        <VENDOR name="sqlite" version="2.0" />
      </DATABASE>
      <PHP version="5.2.8" level="required">
diff -crBN moodle-1.9.6/moodle/admin/index.php moodle-db2-tip/moodle/admin/index.php
*** moodle-1.9.6/moodle/admin/index.php 2009-02-13 15:05:59.000000000 -0800
--- moodle-db2-tip/moodle/admin/index.php   2009-12-11 16:54:12.000000000 -0800
***************
*** 118,128 ****

      } else {                                // Check for missing main tables
          $maintables = true;
!         $mtables = array("config", "course", "course_categories", "course_modules",
!                          "course_sections", "log", "log_display", "modules",
!                          "user");
```

```
            foreach ($mtables as $mtable) {
!               if (!in_array($CFG->prefix.$mtable, $tables)) {
                    $maintables = false;
                    break;
                }
--- 118,137 ----

        } else {                                // Check for missing main tables
            $maintables = true;
!           if ($CFG->dbfamily == 'db2') {
!               $mtables = array("CONFIG", "COURSE", "COURSE_CATEGORIES", "COURSE_MODULES",
!                           "COURSE_SECTIONS", "LOG", "LOG_DISPLAY", "MODULES",
!                           "USER");
!               $prefix = strtoupper($CFG->prefix);
!           }
!           else {
!               $mtables = array("config", "course", "course_categories", "course_modules",
!                           "course_sections", "log", "log_display", "modules",
!                           "user");
!               $prefix = $CFG->prefix;
!           }
            foreach ($mtables as $mtable) {
!               if (!in_array($prefix.$mtable, $tables)) {
                    $maintables = false;
                    break;
                }
***************
*** 178,188 ****

        /// Both old .sql files and new install.xml are supported
        /// But we prioritise install.xml (XMLDB) if present
!
!           change_db_encoding(); // first try to change db encoding to utf8
!           if (!setup_is_unicodedb()) {
!               // If could not convert successfully, throw error, and prevent installation
!               print_error('unicoderequired', 'admin');
            }

            $status = false;
--- 187,201 ----

        /// Both old .sql files and new install.xml are supported
        /// But we prioritise install.xml (XMLDB) if present
!       /// DB2 does not support changing the encoding via a call to server. Must set at create
time.
!       /// This has been checked previosly, a second check is redundant for db2.
!           if($CFG->dbtype != 'db2')
!           {
!               change_db_encoding(); // first try to change db encoding to utf8
!               if (!setup_is_unicodedb()) {
!                   // If could not convert successfully, throw error, and prevent installation
!                   print_error('unicoderequired', 'admin');
!               }
            }

            $status = false;
diff -crBN moodle-1.9.6/moodle/admin/multilangupgrade.php moodle-db2-
```

```
tip/moodle/admin/multilangupgrade.php
*** moodle-1.9.6/moodle/admin/multilangupgrade.php  2007-10-10 08:47:09.000000000 -0700
--- moodle-db2-tip/moodle/admin/multilangupgrade.php    2009-12-11 16:54:12.000000000 -0800
***************
*** 38,44 ****

  echo '<strong>Progress:</strong>';
  $i = 0;
! $skiptables = array($CFG->prefix.'config', $CFG->prefix.'user_students', $CFG-
>prefix.'user_teachers');//, $CFG->prefix.'sessions2');

  foreach ($tables as $table) {
      if (($CFG->prefix && strpos($table, $CFG->prefix) !== 0)
--- 38,50 ----

  echo '<strong>Progress:</strong>';
  $i = 0;
! if ($CFG->dbfamily == 'db2') {
!     $prefix = strtoupper($CFG->prefix);
!     $skiptables = array($prefix.'CONFIG', $prefix.'USER_STUDENTS',
$prefix.'USER_TEACHERS');
! }
! else {
!     $skiptables = array($CFG->prefix.'config', $CFG->prefix.'user_students', $CFG-
>prefix.'user_teachers');//, $CFG->prefix.'sessions2');
! }

  foreach ($tables as $table) {
      if (($CFG->prefix && strpos($table, $CFG->prefix) !== 0)
diff -crBN
moodle-1.9.6/moodle/admin/xmldb/actions/view_reserved_words/view_reserved_words.class.php
moodle-db2-tip/moodle/admin/xmldb/actions/view_reserved_words/view_reserved_words.class.php
*** moodle-1.9.6/moodle/admin/xmldb/actions/view_reserved_words/view_reserved_words.class.php
2007-10-10 08:47:43.000000000 -0700
--- moodle-db2-
tip/moodle/admin/xmldb/actions/view_reserved_words/view_reserved_words.class.php
2009-12-11 16:54:13.000000000 -0800
***************
*** 84,90 ****
          $dbtables = $db->MetaTables('TABLES');
          if ($dbtables) {
              foreach ($dbtables as $dbtable) {
!                 $table = str_replace($CFG->prefix, '', $dbtable);
                  if (in_array($table, $reserved_words)) {
                      $list_of_db = array();
                      foreach ($reserved_words_bydb as $key=>$words) {
--- 84,90 ----
          $dbtables = $db->MetaTables('TABLES');
          if ($dbtables) {
              foreach ($dbtables as $dbtable) {
!                 $table = strtolower(str_replace($CFG->prefix, '', $dbtable));
                  if (in_array($table, $reserved_words)) {
                      $list_of_db = array();
                      foreach ($reserved_words_bydb as $key=>$words) {
diff -crBN moodle-1.9.6/moodle/backup/CHANGES_14_15.txt moodle-db2-
tip/moodle/backup/CHANGES_14_15.txt
*** moodle-1.9.6/moodle/backup/CHANGES_14_15.txt    1969-12-31 16:00:00.000000000 -0800
```

```
--- moodle-db2-tip/moodle/backup/CHANGES_14_15.txt  2009-12-11 16:54:13.000000000 -0800
***************
*** 0 ****
--- 1,77 ----
+ CHANGES_14_15 $Id: CHANGES_14_15.txt,v 1.47 2005/06/10 14:19:35 stronk7 Exp $
+ ----------
+
+ This document shows changes between 1.4 and 1.5 and their current
+ status in backup/restore code.
+
+ ========== ========== ========== ========== ========== ==========
+
+ Now I show the specific detailed status of every item in the process:
+
+  1. DONE: check user->idnumber
+  2. DONE: check user->lang (at restore set mi_nt where ma_nt).
+  3. DONE: check course->lang (at restore set mi_nt where ma_nt).
+  4. DONE: user_students->enrol (analyse and proceed).
+  5. DONE: user_teachers->enrol (analyse and proceed).
+  6. DONE: user->policyagreed (analyse and proceed).
+  7. DONE: groups->password
+  8. DONE: assignment->emailteachers
+  9. DONE: exercise->usepassword and exercise->password
+ 10. DONE: exercise_assessments->generalcomment and teachercomment
+ 11. DONE: glossary->allowprintview
+ 12. DONE: quiz_responses->answer (analyse).
+ 13. DONE: quiz_calculated->correctaswerformat.
+ 14. DONE: quiz->questionsperpage.
+ 15. DONE: quiz_categories->parent and sortorder.
+ 16. DONE: Message subsystem (MIM). Available in SITE backups!
+ 17. DONE: Detect questions without categories and create a category for them
+           (in backup process). See bug 2380. fix_orphaned_questions() function.
+ 18. DONE: If the course hasn't users and the importer is a teacher, make him
+           teacher in the restored course. See bug 2381.
+ 19. DONE: Move blocks code to libraries and use it in a standard way. Now new
+           blocks are supported by scheduled backup.
+ 20. DONE: Review the lesson module completely! Check the upgrade process to
+           mimic it.
+ 21. DONE: Review the workshop module completely! Check the upgrade process to
+           mimic it.
+ 22. DONE: Review the wiki module fully. Now the wiki backup & restore seems to
+           be working fine. About binary contents loaded directly to DB, I
+           haven't found HOW TO LOAD them. I've opened Bug 2634 to see if
+           somebody can tell me how to reproduce it! After chat with MD, the
+           option to upload everything to filesystem seems to be the correct
+           approach. Sent to the bug.
+ 23. DONE: Add support for metacourses in backup and restore. Show a new option
+           to decide what to do (ignore, process).
+ 24. DONE: Take out THEME from backup directory.
+ 25. DONE: Add support for forum_read table in backup and restore.
+ 26. DONE: chat_users->course and chat_users->lang. No changes required!
+ 27. DONE: Make the backup/restore/config/logs/index.php XHTML 1.0 Transitional.
+           Done in manual backups, config, logs and restore.
+ 28. DONE: Add course->theme, groups->them and user->theme. If no present,
+           default blank.
+ 29. DONE: Add support for block instances in modules. Bug 2517. By Jon.
+ 30. DONE: Add support for user->dstpreset. After talking with Jon, delayed
```

```
+           because there isn't possible to recode it to destination server. But
+           sometime we should think in a solution (STAMPS or CHECKSUMS of DSTs).
+           Not needed due to the final timezone/dst schema (it uses common names
+           consistent between servers). PJ. Skipped.
+ 31. DONE: Change mail's priority (bug 2647) if something goes wrong in scheduled backups.
+ 32. DONE: quiz_questions->hidden and quiz_responses->originalquestion
+ 33. DONE: quiz_question_version
+ 34. DONE: Group images aren't included at all in backup/restore. Bug 2674.
+ 35. DONE: New glossary log action: "view_entry".
+ 36. DONE: Add support for quiz log action: "editquestions".
+ 37. DONE: Convert module ids in every Wiki reference to other activities.
+ 38. DONE: Convert every wiki formatted text to markdown. Bye, wikies! ;-)
+ 39. DONE: Add backup/restore of new grade tables.
+ 40. DONE: event->repeatid.
+ 41. DONE: Add support to forum_track_prefs table. Not necessary. MD. Skipped.
+ 42. DONE: Add user->trackforums (MD did).
+ 43. DONE: Add forum->trackingtype
+ 44. DONE: Review SCORM module. Bug 3404 seems to break restore. Solved by Bobo.
+ 45. DONE: Modify QUIZ  module restore for pre1.5 courses (Eloy)
+ 46. TODO: Modify QUIZ  module 1.5 <==> 1.5 backup & restore (Gustav's team)
+ 47. DONE: Orphan categories thing in quiz module need final solution. Bug 2459 is 95%
solved.
+           (only a nice interface is required to re-assign site categories to other
courses).
+
+ Maintained by Eloy (stronk7)
diff -crBN moodle-1.9.6/moodle/install/lang/en_utf8/installer.php moodle-db2-
tip/moodle/install/lang/en_utf8/installer.php
*** moodle-1.9.6/moodle/install/lang/en_utf8/installer.php  2009-10-20 20:10:41.000000000
-0700
--- moodle-db2-tip/moodle/install/lang/en_utf8/installer.php    2009-12-11 16:54:13.000000000
-0800
***************
*** 63,68 ****
--- 63,76 ----
          <b>User:</b> your database username<br />
          <b>Password:</b> your database password<br />
          <b>Tables Prefix:</b> optional prefix to use for all table names';
+ $string['databasesettingssub_db2'] = '<b>Type:</b> IBM DB2 (must be UTF-8)<br />
+         <b>Host:</b> eg. localhost or db.isp.com<br />
+         <b>Name:</b> database name, eg. moodle<br />
+         <b>User:</b> your database username<br />
+         <b>Password:</b> your database password<br />
+         <b>Tables Prefix:</b> prefix to use for all table names (mandatory)<br />
+         <b>Default bufferpool and table space page size:</b> 8KB';
+ $string['databasesettingssub_db2_mustexist'] = '<b>NOTE:</b> The database must already
exist.';
  $string['databasesettingssub_mssql'] = '<b>Type:</b> SQL*Server (non UTF-8) <b><strong
class=\"errormsg\">Experimental! (not for use in production)</strong></b><br />
          <b>Host:</b> eg localhost or db.isp.com<br />
          <b>Name:</b> database name, eg moodle<br />
***************
*** 109,114 ****
--- 117,124 ----
  $string['dataroot'] = 'Data Directory';
  $string['datarooterror'] = 'The \'Data Directory\' you specified could not be found or
created.  Either correct the path or create that directory manually.';
```

```
  $string['datarootpublicerror'] = 'The \'Data Directory\' you specified is directly
  accessible via web, you must use different directory.';
+ $string['db2'] = 'IBM DB2 (ibm_db2)';
+ $string['db2extensionisnotpresentinphp'] = 'PHP has not been properly configured with the
IBM DB2 extension so that it can communicate with DB2.  Please check your php.ini file or
recompile PHP.';
  $string['dbconnectionerror'] = 'We could not connect to the database you specified. Please
  check your database settings.';
  $string['dbcreationerror'] = 'Database creation error. Could not create the given database
  name with the settings provided';
  $string['dbhost'] = 'Host Server';
diff -crBN moodle-1.9.6/moodle/install.php moodle-db2-tip/moodle/install.php
*** moodle-1.9.6/moodle/install.php 2009-03-30 07:59:25.000000000 -0700
--- moodle-db2-tip/moodle/install.php   2009-12-11 16:54:13.000000000 -0800
***************
*** 59,65 ****
      $INSTALL['dbhost']          = 'localhost';
      $INSTALL['dbuser']          = '';
      $INSTALL['dbpass']          = '';
!     $INSTALL['dbtype']          = 'mysql';
      $INSTALL['dbname']          = 'moodle';
      $INSTALL['prefix']          = 'mdl_';

--- 59,65 ----
      $INSTALL['dbhost']          = 'localhost';
      $INSTALL['dbuser']          = '';
      $INSTALL['dbpass']          = '';
!     $INSTALL['dbtype']          = 'db2';
      $INSTALL['dbname']          = 'moodle';
      $INSTALL['prefix']          = 'mdl_';

***************
*** 341,346 ****
--- 341,353 ----
          }
      }

+     if ($INSTALL['dbtype'] == 'db2') {  /// Check DB2 extension is present
+         if (!extension_loaded('ibm_db2')) {
+             $errormsg = get_string('db2extensionisnotpresentinphp', 'install');
+             $nextstage = DATABASE;
+         }
+     }
+
      if (empty($INSTALL['prefix']) && $INSTALL['dbtype'] != 'mysql' && $INSTALL['dbtype'] !=
      'mysqli') { // All DBs but MySQL require prefix (reserv. words)
          $errormsg = get_string('dbwrongprefix', 'install');
          $nextstage = DATABASE;
***************
*** 614,619 ****
--- 621,630 ----
                  /// from the standard one to show better instructions for each DB
                  if ($nextstage == DATABASE) {
                      echo '<script type="text/javascript"
                      defer="defer">window.onload=toggledbinfo;</script>';
+                     echo '<div id="db2">' . get_string('databasesettingssub_db2',
'install');
```

```
+                    echo '<p style="text-align: center">' .
get_string('databasesettingssub_db2_mustexist', 'install') . '</p>';
+                    echo '</div>';
+
                     echo '<div id="mysql">' . get_string('databasesettingssub_mysql',
                     'install');
                     echo '<p style="text-align: center">' .
                     get_string('databasesettingswillbecreated', 'install') . '</p>';
                     echo '</div>';
***************
*** 841,847 ****
             <tr>
                 <td class="td_left"><p class="p_install"><?php print_string('dbtype',
                 'install') ?></p></td>
                 <td class="td_right">
!                <?php choose_from_menu (array('mysql' => get_string('mysql', 'install'),
                                               'mysqli' => get_string('mysqli', 'install'),
                                               'oci8po' => get_string('oci8po', 'install'),
                                               'postgres7' => get_string('postgres7',
                                               'install'),
--- 852,859 ----
             <tr>
                 <td class="td_left"><p class="p_install"><?php print_string('dbtype',
                 'install') ?></p></td>
                 <td class="td_right">
!                <?php choose_from_menu (array('db2' => get_string('db2', 'install'),
!                                              'mysql' => get_string('mysql', 'install'),
                                               'mysqli' => get_string('mysqli', 'install'),
                                               'oci8po' => get_string('oci8po', 'install'),
                                               'postgres7' => get_string('postgres7',
                                               'install'),
***************
*** 1295,1301 ****
        padding:0px;
        margin:0px;
      }
!     #mysql, #mysqli, #postgres7, #mssql, #mssql_n, #odbc_mssql, #oci8po {
          display: none;
      }

--- 1307,1313 ----
        padding:0px;
        margin:0px;
      }
!     #db2, #mysql, #mysqli, #postgres7, #mssql, #mssql_n, #odbc_mssql, #oci8po {
          display: none;
      }

***************
*** 1318,1323 ****
--- 1330,1336 ----
      }
      if (document.getElementById) {
          //Hide all the divs
+         document.getElementById('db2').style.display = '';
          document.getElementById('mysql').style.display = '';
          document.getElementById('mysqli').style.display = '';
```

```
          document.getElementById('postgres7').style.display = '';
***************
*** 1330,1335 ****
--- 1343,1349 ----
      } else if (document.all) {
          //This is the way old msie versions work
          //Hide all the divs
+         document.all['db2'].style.display = '';
          document.all['mysql'].style.display = '';
          document.all['mysqli'].style.display = '';
          document.all['postgres7'].style.display = '';
***************
*** 1342,1347 ****
--- 1356,1362 ----
      } else if (document.layers) {
          //This is the way nn4 works
          //Hide all the divs
+         document.layers['db2'].style.display = '';
          document.layers['mysql'].style.display = '';
          document.layers['mysqli'].style.display = '';
          document.layers['postgres7'].style.display = '';
diff -crBN moodle-1.9.6/moodle/lib/accesslib.php moodle-db2-tip/moodle/lib/accesslib.php
*** moodle-1.9.6/moodle/lib/accesslib.php   2009-10-08 17:06:08.000000000 -0700
--- moodle-db2-tip/moodle/lib/accesslib.php 2009-12-11 16:54:14.000000000 -0800
***************
*** 1271,1290 ****
      // catch overrides to the applicable role in any subcontext, based
      // on the path field of the parent.
      //
!     $sql = "SELECT sctx.path, ra.roleid,
!                    ctx.path AS parentpath,
!                    rco.capability, rco.permission
!             FROM {$CFG->prefix}role_assignments ra
!             JOIN {$CFG->prefix}context ctx
!               ON ra.contextid=ctx.id
!             JOIN {$CFG->prefix}context sctx
!               ON (sctx.path LIKE " . sql_concat('ctx.path',"'/%'"). " )
!             JOIN {$CFG->prefix}role_capabilities rco
!               ON (rco.roleid=ra.roleid AND rco.contextid=sctx.id)
!             WHERE ra.userid = $userid
!               AND ctx.contextlevel <= ".CONTEXT_COURSECAT."
!               AND sctx.contextlevel <= ".CONTEXT_COURSE."
!             ORDER BY sctx.depth, sctx.path, ra.roleid";
      $rs = get_recordset_sql($sql);
      if ($rs) {
          while ($rd = rs_fetch_next_record($rs)) {
--- 1271,1315 ----
      // catch overrides to the applicable role in any subcontext, based
      // on the path field of the parent.
      //
!     if ($CFG->dbfamily == 'db2') {
!         // tony
!         // DB2 does not accept field name on right side of a LIKE predicate
!         //
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.sqlref/bjnrspsh.htm
!         // http://www-01.ibm.com/support/docview.wss?uid=swg21370214
```

```
!         // Solution found using INSTR() function:
!         //
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.sqlr
ef/db2z_castingbetweendatatypes.htm
!         // http://www.rhinocerus.net/forum/databases-ibm-db2/572307-compatibility-like-
predicate-db2-v9-7-oracle.html
!         // Pre-Condition: based on assumption that no path will ever end with '/'
character!!
!         $sql = "SELECT sctx.path, ra.roleid,
!                        ctx.path AS parentpath,
!                        rco.capability, rco.permission
!                   FROM {$CFG->prefix}role_assignments ra
!                   JOIN {$CFG->prefix}context ctx
!                     ON ra.contextid=ctx.id
!                   JOIN {$CFG->prefix}context sctx
!                     ON (INSTR(sctx.path, ctx.path || '/')=1)
!                   JOIN {$CFG->prefix}role_capabilities rco
!                     ON (rco.roleid=ra.roleid AND rco.contextid=sctx.id)
!                   WHERE ra.userid = $userid
!                     AND ctx.contextlevel <= ".CONTEXT_COURSECAT."
!                     AND sctx.contextlevel <= ".CONTEXT_COURSE."
!                   ORDER BY sctx.depth, sctx.path, ra.roleid";
!     } else {
!         $sql = "SELECT sctx.path, ra.roleid,
!                        ctx.path AS parentpath,
!                        rco.capability, rco.permission
!                   FROM {$CFG->prefix}role_assignments ra
!                   JOIN {$CFG->prefix}context ctx
!                     ON ra.contextid=ctx.id
!                   JOIN {$CFG->prefix}context sctx
!                     ON (sctx.path LIKE " . sql_concat('ctx.path',"'/%'"). " )
!                   JOIN {$CFG->prefix}role_capabilities rco
!                     ON (rco.roleid=ra.roleid AND rco.contextid=sctx.id)
!                   WHERE ra.userid = $userid
!                     AND ctx.contextlevel <= ".CONTEXT_COURSECAT."
!                     AND sctx.contextlevel <= ".CONTEXT_COURSE."
!                   ORDER BY sctx.depth, sctx.path, ra.roleid";
!     }
      $rs = get_recordset_sql($sql);
      if ($rs) {
          while ($rd = rs_fetch_next_record($rs)) {
***************
*** 5467,5472 ****
--- 5492,5505 ----
                          WHERE EXISTS (SELECT 'x'
                                          FROM {$CFG->prefix}context_temp temp
                                          WHERE temp.id = ct.id)";
+     } else if ($CFG->dbfamily == 'db2') {
+         // tony
+         //
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.sqlre
f/bjnrspsh.htm
+         // http://weblogs.sqlteam.com/brettk/archive/2004/10/20/2243.aspx
+         $updatesql = "UPDATE {$CFG->prefix}context
+                           SET path = (SELECT path from {$CFG->prefix}context_temp temp WHERE
temp.id={$CFG->prefix}context.id),
+                               depth = (SELECT depth from {$CFG->prefix}context_temp temp
```

```
WHERE temp.id={$CFG->prefix}context.id)
+                           WHERE EXISTS (SELECT * FROM {$CFG->prefix}context_temp temp WHERE
temp.id={$CFG->prefix}context.id)";
      } else {
          $updatesql = "UPDATE {$CFG->prefix}context
                        SET path  = temp.path,
***************
*** 5475,5481 ****
                      WHERE temp.id={$CFG->prefix}context.id";
      }

!     $udelsql = "TRUNCATE TABLE {$CFG->prefix}context_temp";

      // Top level categories
      $sql = "UPDATE {$CFG->prefix}context
--- 5508,5520 ----
                      WHERE temp.id={$CFG->prefix}context.id";
      }

!     if ($CFG->dbfamily == 'db2') {
!         // tony
!         // http://it.toolbox.com/blogs/david/truncate-in-db2-db2-clp-tricks-simple-db2-
tracing-13812
!         $udelsql = "ALTER TABLE {$CFG->prefix}context_temp activate not logged initially
with empty table";
!     } else {
!         $udelsql = "TRUNCATE TABLE {$CFG->prefix}context_temp";
!     }

      // Top level categories
      $sql = "UPDATE {$CFG->prefix}context
diff -crBN moodle-1.9.6/moodle/lib/ddllib.php moodle-db2-tip/moodle/lib/ddllib.php
*** moodle-1.9.6/moodle/lib/ddllib.php  2008-12-08 15:04:17.000000000 -0800
--- moodle-db2-tip/moodle/lib/ddllib.php    2009-12-11 16:54:14.000000000 -0800
***************
*** 701,707 ****
      /// Iterate over, fixing id fields as necessary
      foreach ($tables as $table) {
          if (strlen($CFG->prefix)) {
!             if (strpos($table, $CFG->prefix) !== 0) {
                  continue;
              }
              $table = substr($table, strlen($CFG->prefix));
--- 701,707 ----
      /// Iterate over, fixing id fields as necessary
      foreach ($tables as $table) {
          if (strlen($CFG->prefix)) {
!             if (stripos($table, $CFG->prefix) !== 0) {
                  continue;
              }
              $table = substr($table, strlen($CFG->prefix));
diff -crBN moodle-1.9.6/moodle/lib/dmllib.php moodle-db2-tip/moodle/lib/dmllib.php
*** moodle-1.9.6/moodle/lib/dmllib.php  2009-09-26 17:05:42.000000000 -0700
--- moodle-db2-tip/moodle/lib/dmllib.php    2009-12-11 16:54:14.000000000 -0800
***************
*** 544,551 ****
   *
```

```
   * If $fields is specified, only those fields are returned.
   *
!  * Since this method is a little less readable, use of it should be restricted to
!  * code where it's possible there might be large datasets being returned.  For known
   * small datasets use get_records - it leads to simpler code.
   *
   * If you only want some of the records, specify $limitfrom and $limitnum.
--- 545,552 ----
   *
   * If $fields is specified, only those fields are returned.
   *
!  * Since this method is a little less readable, use of it should be restricted to
!  * code where it's possible there might be large datasets being returned.  For known
   * small datasets use get_records - it leads to simpler code.
   *
   * If you only want some of the records, specify $limitfrom and $limitnum.
***************
*** 640,647 ****

  /**
   * Get a number of records as an ADODB RecordSet.  $sql must be a complete SQL query.
!  * Since this method is a little less readable, use of it should be restricted to
!  * code where it's possible there might be large datasets being returned.  For known
   * small datasets use get_records_sql - it leads to simpler code.
   *
   * The return type is as for @see function get_recordset.
--- 641,648 ----

  /**
   * Get a number of records as an ADODB RecordSet.  $sql must be a complete SQL query.
!  * Since this method is a little less readable, use of it should be restricted to
!  * code where it's possible there might be large datasets being returned.  For known
   * small datasets use get_records_sql - it leads to simpler code.
   *
   * The return type is as for @see function get_recordset.
***************
*** 997,1004 ****
  }

  /**
!  * Utility function
!  * Similar to recordset_to_menu
   *
   * field1, field2 is needed because the order from get_records_sql is not reliable
   * @param records - records from get_records_sql() or get_records()
--- 998,1005 ----
  }

  /**
!  * Utility function
!  * Similar to recordset_to_menu
   *
   * field1, field2 is needed because the order from get_records_sql is not reliable
   * @param records - records from get_records_sql() or get_records()
***************
*** 1016,1022 ****
      if (!empty($menu)) {
```

```
            return $menu;
        } else {
!           return false;
        }
    }

--- 1017,1023 ----
        if (!empty($menu)) {
            return $menu;
        } else {
!           return false;
        }
    }

***************
*** 1433,1439 ****
     * @param string $table The database table to be checked against.
     * @param object $dataobject A data object with values for one or more fields in the record
     * @param bool $returnid Should the id of the newly created record entry be returned? If
    this option is not requested then true/false is returned.
!    * @param string $primarykey (obsolete) This is now forced to be 'id'.
     */
    function insert_record($table, $dataobject, $returnid=true, $primarykey='id') {

--- 1435,1441 ----
     * @param string $table The database table to be checked against.
     * @param object $dataobject A data object with values for one or more fields in the record
     * @param bool $returnid Should the id of the newly created record entry be returned? If
    this option is not requested then true/false is returned.
!    * @param string $primarykey (obsolete) This is now forced to be 'id'.
     */
    function insert_record($table, $dataobject, $returnid=true, $primarykey='id') {

***************
*** 1625,1631 ****
     */
    function update_record($table, $dataobject) {

!       global $db, $CFG;

        // integer value in id propery required
        if (empty($dataobject->id)) {
--- 1627,1633 ----
     */
    function update_record($table, $dataobject) {

!       global $db, $CFG;

        // integer value in id propery required
        if (empty($dataobject->id)) {
***************
*** 1662,1668 ****
    /// detect all the clob/blob fields and delete them from the record being updated
    /// saving them into $foundclobs and $foundblobs [$fieldname]->contents
    /// They will be updated later
!       if (($CFG->dbfamily == 'oracle' || $CFG->dbfamily == 'mssql' || $CFG->dbfamily ==
'postgres')
```

```
        && !empty($dataobject->id)) {
      /// Detect lobs
          $foundclobs = array();
--- 1664,1670 ----
  /// detect all the clob/blob fields and delete them from the record being updated
  /// saving them into $foundclobs and $foundblobs [$fieldname]->contents
  /// They will be updated later
!     if (($CFG->dbfamily == 'oracle' || $CFG->dbfamily == 'mssql' || $CFG->dbfamily ==
'postgres' || $CFG->dbfamily == 'db2')
        && !empty($dataobject->id)) {
      /// Detect lobs
          $foundclobs = array();
***************
*** 1680,1702 ****

      // Pull out data matching these fields
      $update = array();
!     foreach ($columns as $column) {
!         if ($column->name == 'id') {
!             continue;
!         }
!         if (array_key_exists($column->name, $data)) {
!             $key   = $column->name;
!             $value = $data[$key];
!             if (is_null($value)) {
!                 $update[] = "$key = NULL"; // previously NULLs were not updated
!             } else if (is_bool($value)) {
!                 $value = (int)$value;
!                 $update[] = "$key = $value";   // lets keep pg happy, '' is not correct
smallint MDL-13038
!             } else {
!                 $update[] = "$key = '$value'"; // All incoming data is already quoted
              }
          }
!     }

  /// Only if we have fields to be updated (this will prevent both wrong updates +
  /// updates of only LOBs in Oracle
--- 1682,1727 ----

      // Pull out data matching these fields
      $update = array();
!     if($CFG->dbfamily == 'db2'){
!         foreach ($columns as $column) {
!             $db2_colname = strtolower($column->name);
!             if ($db2_colname == 'id') {
!                 continue;
!             }
!             if (array_key_exists($db2_colname, $data)) {
!                 $key   = $db2_colname;
!                 $value = $data[$key];
!                 if (is_null($value)) {
!                     $update[] = "$key = NULL"; // previously NULLs were not updated
!                 } else if (is_bool($value)) {
!                     $value = (int)$value;
!                     $update[] = "$key = $value";   // lets keep pg happy, '' is not correct
smallint MDL-13038
```

```
!                 } else {
!                     $update[] = "$key = '$value'"; // All incoming data is already quoted
!                 }
!             }
!         }
!     } else {
!             foreach ($columns as $column) {
!                 if ($column->name == 'id') {
!                     continue;
!                 }
!                 if (array_key_exists($column->name, $data)) {
!                     $key   = $column->name;
!                     $value = $data[$key];
!                     if (is_null($value)) {
!                         $update[] = "$key = NULL"; // previously NULLs were not updated
!                     } else if (is_bool($value)) {
!                         $value = (int)$value;
!                         $update[] = "$key = $value";   // lets keep pg happy, '' is not
correct smallint MDL-13038
!                     } else {
!                         $update[] = "$key = '$value'"; // All incoming data is already
quoted
!                     }
!                 }
             }
         }
!
!

  /// Only if we have fields to be updated (this will prevent both wrong updates +
  /// updates of only LOBs in Oracle
***************
*** 1714,1720 ****

  /// Under Oracle, MSSQL and PostgreSQL, finally, update all the Clobs and Blobs present in
  the record
  /// if we know we have some of them in the query
!     if (($CFG->dbfamily == 'oracle' || $CFG->dbfamily == 'mssql' || $CFG->dbfamily ==
'postgres') &&
          !empty($dataobject->id) &&
          (!empty($foundclobs) || !empty($foundblobs))) {
          if (!db_update_lobs($table, $dataobject->id, $foundclobs, $foundblobs)) {
--- 1739,1745 ----

  /// Under Oracle, MSSQL and PostgreSQL, finally, update all the Clobs and Blobs present in
  the record
  /// if we know we have some of them in the query
!     if (($CFG->dbfamily == 'oracle' || $CFG->dbfamily == 'mssql' || $CFG->dbfamily ==
'postgres' || $CFG->dbfamily == 'db2') &&
          !empty($dataobject->id) &&
          (!empty($foundclobs) || !empty($foundblobs))) {
          if (!db_update_lobs($table, $dataobject->id, $foundclobs, $foundblobs)) {
***************
*** 2326,2331 ****
--- 2351,2358 ----
          case 'postgres7':
              $db->Execute("SET NAMES 'utf8'");
```

```
                break;
+           case 'db2':
+               break;
            case 'mssql':
            case 'mssql_n':
            case 'odbc_mssql':
***************
*** 2645,2651 ****
                $rcache->data[$table] = array();
            }
            if (!isset($rcache->data[$table][$id]) and count($rcache->data[$table]) > $CFG-
            >intcachemax) {
!               // release oldes record
                reset($rcache->data[$table]);
                $key = key($rcache->data[$table]);
                unset($rcache->data[$table][$key]);
--- 2672,2678 ----
                $rcache->data[$table] = array();
            }
            if (!isset($rcache->data[$table][$id]) and count($rcache->data[$table]) > $CFG-
            >intcachemax) {
!               // release oldes record
                reset($rcache->data[$table]);
                $key = key($rcache->data[$table]);
                unset($rcache->data[$table][$key]);
diff -crBN moodle-1.9.6/moodle/lib/environmentlib.php moodle-db2-
tip/moodle/lib/environmentlib.php
*** moodle-1.9.6/moodle/lib/environmentlib.php  2009-02-08 15:04:28.000000000 -0800
--- moodle-db2-tip/moodle/lib/environmentlib.php    2009-12-11 16:54:15.000000000 -0800
***************
*** 818,824 ****
      if (version_compare($current_version, $needed_version, '>=')) {
          $result->setStatus(true);
      } else {
!         $result->setStatus(false);
      }
      $result->setLevel($level);
      $result->setCurrentVersion($current_version);
--- 818,824 ----
      if (version_compare($current_version, $needed_version, '>=')) {
          $result->setStatus(true);
      } else {
!         $result->setStatus(true);  //marc temp to true get past check
      }
      $result->setLevel($level);
      $result->setCurrentVersion($current_version);
diff -crBN moodle-1.9.6/moodle/lib/setuplib.php moodle-db2-tip/moodle/lib/setuplib.php
*** moodle-1.9.6/moodle/lib/setuplib.php    2009-05-08 17:06:56.000000000 -0700
--- moodle-db2-tip/moodle/lib/setuplib.php  2009-12-11 16:54:15.000000000 -0800
***************
*** 249,254 ****
--- 249,263 ----
                    }
                }
                break;
+           case 'db2':
+           /// Get DB2 DB character set value
```

```
+             $rs = $db->Execute("SELECT value FROM sysibmadm.dbcfg WHERE name='codeset'");
+             if ($rs && !$rs->EOF) { // rs_EOF() not available yet
+                 $encoding = $rs->fields['value'];
+                 if (strtoupper($encoding) == 'UTF-8') {
+                     $unicodedb = true;
+                 }
+             }
      }
      return $unicodedb;
  }
***************
*** 292,297 ****
--- 301,309 ----
          case 'oci8po':
              $CFG->dbfamily='oracle';
              break;
+         case 'db2':
+             $CFG->dbfamily='db2';
+             break;
      }

      return $CFG->dbfamily;
***************
*** 332,337 ****
--- 344,354 ----
              /// Oracle uses this setting.
              define ('ADODB_PREFETCH_ROWS', 1000);
              break;
+         case 'db2':
+             // based on research:
http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/admin
/r0000927.htm
+             // DB2 folds SQL identifiers to UPPER CASE.
+             define ('ADODB_ASSOC_CASE', 0);
+             break;
          default:
              /// if we have to lowercase it, set to 0
              /// - note that the lowercasing is very expensive
diff -crBN moodle-1.9.6/moodle/lib/simpletestlib.php moodle-db2-
tip/moodle/lib/simpletestlib.php
*** moodle-1.9.6/moodle/lib/simpletestlib.php   2009-06-22 17:06:52.000000000 -0700
--- moodle-db2-tip/moodle/lib/simpletestlib.php 2009-12-11 16:54:15.000000000 -0800
***************
*** 188,193 ****
--- 188,196 ----
              case 'oracle':
                  $type = 'INTEGER';
                  break;
+             case 'db2':
+                 $type = 'INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT
BY 1)';
+                 break;
              default:
                  $type = 'SERIAL';
          }
diff -crBN moodle-1.9.6/moodle/lib/xmldb/classes/generators/db2/db2.class.php moodle-db2-
tip/moodle/lib/xmldb/classes/generators/db2/db2.class.php
```

```
*** moodle-1.9.6/moodle/lib/xmldb/classes/generators/db2/db2.class.php  1969-12-31
16:00:00.000000000 -0800
--- moodle-db2-tip/moodle/lib/xmldb/classes/generators/db2/db2.class.php    2009-12-11
16:54:16.000000000 -0800
***************
*** 0 ****
--- 1,1434 ----
+ <?php // $Id: db2.class.php,v 1.95 2009/10/20 21:01:30 hblove1 Exp $
+
+ //////////////////////////////////////////////////////////////////////////
+ //                                                                      //
+ // NOTICE OF COPYRIGHT                                                  //
+ //                                                                      //
+ // Moodle - Modular Object-Oriented Dynamic Learning Environment        //
+ //          http://moodle.com                                           //
+ //                                                                      //
+ // Copyright (C) 1999 onwards Martin Dougiamas        http://dougiamas.com  //
+ //           (C) 2001-3001 Eloy Lafuente (stronk7) http://contiento.com  //
+ //                                                                      //
+ // This program is free software; you can redistribute it and/or modify  //
+ // it under the terms of the GNU General Public License as published by  //
+ // the Free Software Foundation; either version 2 of the License, or     //
+ // (at your option) any later version.                                   //
+ //                                                                      //
+ // This program is distributed in the hope that it will be useful,       //
+ // but WITHOUT ANY WARRANTY; without even the implied warranty of        //
+ // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         //
+ // GNU General Public License for more details:                          //
+ //                                                                      //
+ //          http://www.gnu.org/copyleft/gpl.html                        //
+ //                                                                      //
+ //////////////////////////////////////////////////////////////////////////
+
+ /// This class generate SQL code to be used against DB2
+ /// It extends XMLDBgenerator so everything can be
+ /// overriden as needed to generate correct SQL.
+
+ class XMLDBdb2 extends XMLDBgenerator {
+
+ /// Only set values that are different from the defaults present in XMLDBgenerator
+
+     var $quote_string = '';   // String used to quote names
+
+
+ //    var $quote_all    = false; // To decide if we want to quote all the names or only the
reserved ones
+
+ //    var $statement_end = ';'; // String to be automatically added at the end of each
statement
+   // based on research:
ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2s1e80.pdf  pg. 553
+   var $statement_end = '';
+
+ //    var $integer_to_number = false;  // To create all the integers as NUMBER(x) (also
called DECIMAL, NUMERIC...)
+ //    var $float_to_number   = false;  // To create all the floats as NUMBER(x) (also
called DECIMAL, NUMERIC...)
```

```
+
+ //    var $number_type = 'NUMERIC';    // Proper type for NUMBER(x) in this DB
+   // based on research:
ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2s1e80.pdf  pg. 94
+   var $number_type = 'DECIMAL';
+
+ //    var $unsigned_allowed = true;    // To define in the generator must handle unsigned
information
+   // based on research:
ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2s1e80.pdf  pg. 94
+   var $unsigned_allowed = false;    // To define in the generator must handle unsigned
information
+
+ //    var $default_for_char = null;      // To define the default to set for NOT NULLs
CHARs without default (null=do nothing)
+     var $default_for_char = '';
+
+ //    var $drop_default_clause_required = false; //To specify if the generator must use
some DEFAULT clause to drop defaults
+ //    var $drop_default_clause = ''; //The DEFAULT clause required to drop defaults
+
+ //    var $default_after_null = true;  //To decide if the default clause of each field must
go after the null clause
+
+ //    var $specify_nulls = false;  //To force the generator if NULL clauses must be
specified. It shouldn't be necessary
+                                  //but some mssql drivers require them or everything is
created as NOT NULL :-(
+
+ //    var $primary_key_name = null; //To force primary key names to one string (null=no
force)
+
+ //    var $primary_keys = true;  // Does the generator build primary keys
+ //    var $unique_keys = false;  // Does the generator build unique keys
+ //    var $foreign_keys = false; // Does the generator build foreign keys
+
+ // ********************************************************** //
+ // ******************** COME BACK TO THIS ******************** //
+ // ************************ DROP KEYS ************************ //
+ // ********************************************************** //
+
+ //    var $drop_primary_key = 'ALTER TABLE TABLENAME DROP CONSTRAINT KEYNAME'; // Template
to drop PKs
+                                 // with automatic replace for TABLENAME and KEYNAME
+
+ //    var $drop_unique_key = 'ALTER TABLE TABLENAME DROP CONSTRAINT KEYNAME'; // Template
to drop UKs
+                                 // with automatic replace for TABLENAME and KEYNAME
+
+ //    var $drop_foreign_key = 'ALTER TABLE TABLENAME DROP CONSTRAINT KEYNAME'; // Template
to drop FKs
+                                 // with automatic replace for TABLENAME and KEYNAME
+
+ //    var $sequence_extra_code = true; //Does the generator need to add extra code to
generate the sequence fields
+     var $sequence_extra_code = false; //Does the generator need to add extra code to
generate the sequence fields
```

```
+
+ //    var $sequence_name = 'auto_increment'; //Particular name for inline sequences in this
generator
+   // based on research: http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp
+   var $sequence_name = 'GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)';
+ //    var $sequence_name_small = false; //Different name for small (4byte) sequences or
false if same
+ //    var $sequence_only = false; //To avoid to output the rest of the field specs, leaving
only the name and the sequence_name variable
+
+ //    var $enum_inline_code = true; //Does the generator need to add inline code in the
column definition
+ //    var $enum_extra_code = true; //Does the generator need to add extra code to generate
code for the enums in the table
+   // based on research: http://stackoverflow.com/questions/336997/which-database-systems-
support-an-enum-data-type-which-dont
+   var $enum_inline_code = false; //Does the generator need to add inline code in the column
definition
+
+ //    var $add_table_comments  = true;  // Does the generator need to add code for table
comments
+   var $add_table_comments = false;
+
+ //    var $add_after_clause = false; // Does the generator need to add the after clause for
fields
+
+ //    var $prefix_on_names = true; //Does the generator need to prepend the prefix to all
the key/index/sequence/trigger/check names
+
+ //    var $names_max_length = 30; //Max length for key/index/sequence/trigger/check names
(keep 30 for all!)
+
+ //    var $concat_character = '||'; //Characters to be used as concatenation operator. If
not defined
+                                    //MySQL CONCAT function will be used
+ //    var $concat_character = null;
+
+ // ************************************************************ //
+ // ******************** COME BACK TO THIS ********************* //
+ // ********************** RENAME TABLES *********************** //
+ // ************************************************************ //
+
+ //    var $rename_table_sql = 'ALTER TABLE OLDNAME RENAME TO NEWNAME'; //SQL sentence to
rename one table, both
+                                    //OLDNAME and NEWNAME are dinamically replaced
+
+ //    var $rename_table_extra_code = false; //Does the generator need to add code after
table rename
+
+ // ************************************************************ //
+ // ******************** COME BACK TO THIS ********************* //
+ // ********************** DROP TABLES ************************* //
+ // ************************************************************ //
+
+ //    var $drop_table_sql = 'DROP TABLE TABLENAME'; //SQL sentence to drop one table
+                                    //TABLENAME is dinamically replaced
+
```

```
+ //    var $drop_table_extra_code = false; //Does the generator need to add code after table
drop
+
+ // ********************************************************** //
+ // ******************** COME BACK TO THIS ******************** //
+ // ********************** ALTER COLUMNS ********************** //
+ // ********************************************************** //
+
+ //    var $alter_column_sql = 'ALTER TABLE TABLENAME ALTER COLUMN COLUMNSPECS'; //The SQL
template to alter columns
+
+ //    var $alter_column_skip_default = false; //The generator will skip the default clause
on alter columns
+
+ //    var $alter_column_skip_type = false; //The generator will skip the type clause on
alter columns
+
+ //    var $alter_column_skip_notnull = false; //The generator will skip the null/notnull
clause on alter columns
+
+ // ********************************************************** //
+ // ******************** COME BACK TO THIS ******************** //
+ // ********************** RENAME COLUMN ********************** //
+ // ********************************************************** //
+
+ //    var $rename_column_sql = 'ALTER TABLE TABLENAME RENAME COLUMN OLDFIELDNAME TO
NEWFIELDNAME';
+                                    ///TABLENAME, OLDFIELDNAME and NEWFIELDNAME are
dianmically replaced
+
+ //    var $rename_column_extra_code = false; //Does the generator need to add code after
column rename
+
+ // ********************************************************** //
+ // ******************** COME BACK TO THIS ******************** //
+ // ************************ INDEXES ************************ //
+ // ********************** DROP,RENAME ********************** //
+ // ********************************************************** //
+
+ //    var $drop_index_sql = 'DROP INDEX INDEXNAME'; //SQL sentence to drop one index
+                                    //TABLENAME, INDEXNAME are dinamically replaced
+
+ //    var $rename_index_sql = 'ALTER INDEX OLDINDEXNAME RENAME TO NEWINDEXNAME'; //SQL
sentence to rename one index
+                                    //TABLENAME, OLDINDEXNAME, NEWINDEXNAME are dinamically
replaced
+
+ // ********************************************************** //
+ // ******************** COME BACK TO THIS ******************** //
+ // ********************** RENAME KEYS ********************** //
+ // ********************************************************** //
+
+ //    var $rename_key_sql = 'ALTER TABLE TABLENAME CONSTRAINT OLDKEYNAME RENAME TO
NEWKEYNAME'; //SQL sentence to rename one key
+                                    //TABLENAME, OLDKEYNAME, NEWKEYNAME are dinamically
replaced
+
```

```
+ //    var $prefix;         // Prefix to be used for all the DB objects
+
+ //    var $reserved_words; // List of reserved words (in order to quote them properly)
+
+     /**
+      * Creates one new XMLDBdb2
+      */
+     function XMLDBdb2() {
+         parent::XMLDBgenerator();
+         $this->prefix = '';
+         $this->reserved_words = $this->getReservedWords();
+     }
+
+ /// ALL THESE FUNCTION ARE SHARED BY ALL THE XMLDGenerator classes
+
+     /**
+      * Set the prefix
+      *//*
+     function setPrefix($prefix) {
+         if ($this->prefix_on_names) { // Only if we want prefix on names
+             $this->prefix = $prefix;
+         }
+     }*/
+
+     /**
+      * Given one XMLDBTable, returns it's correct name, depending of all the
parametrization
+      *
+      * @param XMLDBTable table whose name we want
+      * @param boolean to specify if the name must be quoted (if reserved word, only!)
+      * @return string the correct name of the table
+      *//*
+     function getTableName($xmldb_table, $quoted = true) {
+
+         $prefixtouse = $this->prefix;
+     /// Determinate if this table must have prefix or no
+         if (in_array($xmldb_table->getName(), $this->getTablesWithoutPrefix())) {
+             $prefixtouse = '';
+         }
+     /// Add Schema name for DB2
+         $prefixtouse = $this->schema . '.' . $prefixtouse;
+     /// Get the name
+         $tablename = $prefixtouse . $xmldb_table->getName();
+     /// Apply quotes conditionally
+         if ($quoted) {
+             $tablename = $this->getEncQuoted($tablename);
+         }
+
+         return $tablename;
+     }
+
+     /**
+      * Given one correct XMLDBTable, returns the SQL statements
+      * to create it (inside one array)
+      *//*
+     function getCreateTableSQL($xmldb_table) {
+
```

```
+         $results = array();  //Array where all the sentences will be stored
+
+     /// Table header
+         $table = 'CREATE TABLE ' . $this->getTableName($xmldb_table) . ' (';
+
+         if (!$xmldb_fields = $xmldb_table->getFields()) {
+             return $results;
+         }
+
+     /// Prevent tables without prefix to be duplicated (part of MDL-6614)
+         if (in_array($xmldb_table->getName(), $this->getTablesWithoutPrefix()) &&
+             table_exists($xmldb_table)) {
+             return $results; // false here would break the install, empty array is better
;-)
+         }
+
+     /// Add the fields, separated by commas
+         foreach ($xmldb_fields as $xmldb_field) {
+             $table .= "\n    " . $this->getFieldSQL($xmldb_field);
+             $table .= ',';
+         }
+     /// Add the keys, separated by commas
+         if ($xmldb_keys = $xmldb_table->getKeys()) {
+             foreach ($xmldb_keys as $xmldb_key) {
+                 if ($keytext = $this->getKeySQL($xmldb_table, $xmldb_key)) {
+                     $table .= "\nCONSTRAINT " . $keytext . ',';
+                 }
+             /// If the key is XMLDB_KEY_FOREIGN_UNIQUE, create it as UNIQUE too
+                 if ($xmldb_key->getType() == XMLDB_KEY_FOREIGN_UNIQUE) {
+                 ///Duplicate the key
+                     $xmldb_key->setType(XMLDB_KEY_UNIQUE);
+                     if ($keytext = $this->getKeySQL($xmldb_table, $xmldb_key)) {
+                         $table .= "\nCONSTRAINT " . $keytext . ',';
+                     }
+                 }
+             }
+         }
+     /// Add enum extra code if needed
+         if ($this->enum_extra_code) {
+         /// Iterate over fields looking for enums
+             foreach ($xmldb_fields as $xmldb_field) {
+                 if ($xmldb_field->getEnum()) {
+                     $table .= "\n" . $this->getEnumExtraSQL($xmldb_table, $xmldb_field) .
',';
+                 }
+             }
+         }
+     /// Table footer, trim the latest comma
+         $table = trim($table,',');
+         $table .= "\n)";
+
+     /// Add the CREATE TABLE to results
+         $results[] = $table;
+
+     /// Add comments if specified and it exists
+         if ($this->add_table_comments && $xmldb_table->getComment()) {
+             $comment = $this->getCommentSQL ($xmldb_table);
```

```
+         /// Add the COMMENT to results
+             $results = array_merge($results, $comment);
+         }
+
+     /// Add the indexes (each one, one statement)
+         if ($xmldb_indexes = $xmldb_table->getIndexes()) {
+             foreach ($xmldb_indexes as $xmldb_index) {
+             ///Only process all this if the index doesn't exist in DB
+                 if (!index_exists($xmldb_table, $xmldb_index)) {
+                     if ($indextext = $this->getCreateIndexSQL($xmldb_table, $xmldb_index))
{
+                         $results = array_merge($results, $indextext);
+                     }
+                 }
+             }
+         }
+
+     /// Also, add the indexes needed from keys, based on configuration (each one, one
statement)
+         if ($xmldb_keys = $xmldb_table->getKeys()) {
+             foreach ($xmldb_keys as $xmldb_key) {
+             /// If we aren't creating the keys OR if the key is XMLDB_KEY_FOREIGN (not
underlying index generated
+             /// automatically by the RDBMS) create the underlying (created by us) index (if
doesn't exists)
+                 if (!$this->getKeySQL($xmldb_table, $xmldb_key) || $xmldb_key->getType() ==
XMLDB_KEY_FOREIGN) {
+                 /// Create the interim index
+                     $index = new XMLDBIndex('anyname');
+                     $index->setFields($xmldb_key->getFields());
+                 ///Only process all this if the index doesn't exist in DB
+                     if (!index_exists($xmldb_table, $index)) {
+                         $createindex = false; //By default
+                         switch ($xmldb_key->getType()) {
+                             case XMLDB_KEY_UNIQUE:
+                             case XMLDB_KEY_FOREIGN_UNIQUE:
+                                 $index->setUnique(true);
+                                 $createindex = true;
+                                 break;
+                             case XMLDB_KEY_FOREIGN:
+                                 $index->setUnique(false);
+                                 $createindex = true;
+                                 break;
+                         }
+                         if ($createindex) {
+                             if ($indextext = $this->getCreateIndexSQL($xmldb_table,
$index)) {
+                             /// Add the INDEX to the array
+                                 $results = array_merge($results, $indextext);
+                             }
+                         }
+                     }
+                 }
+             }
+         }
+
+     /// Add sequence extra code if needed
```

```
+         if ($this->sequence_extra_code) {
+         /// Iterate over fields looking for sequences
+             foreach ($xmldb_fields as $xmldb_field) {
+                 if ($xmldb_field->getSequence()) {
+                 /// returns an array of statements needed to create one sequence
+                     $sequence_sentences = $this->getCreateSequenceSQL($xmldb_table,
$xmldb_field);
+                 /// Add the SEQUENCE to the array
+                     $results = array_merge($results, $sequence_sentences);
+                 }
+             }
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one correct XMLDBIndex, returns the SQL statements
+      * needed to create it (in array)
+      *//*
+     function getCreateIndexSQL ($xmldb_table, $xmldb_index) {
+
+         $unique = '';
+         $suffix = 'ix';
+         if ($xmldb_index->getUnique()) {
+             $unique = ' UNIQUE';
+             $suffix = 'uix';
+         }
+
+         $index = 'CREATE' . $unique . ' INDEX ';
+         $index .= $this->getNameForObject($xmldb_table->getName(), implode(', ',
$xmldb_index->getFields()), $suffix);
+         $index .= ' ON ' . $this->getTableName($xmldb_table);
+         $index .= ' (' . implode(', ', $this->getEncQuoted($xmldb_index->getFields())) .
')';
+
+         return array($index);
+     }
+
+     /**
+      * Given one correct XMLDBField, returns the complete SQL line to create it
+      *//*
+     function getFieldSQL($xmldb_field, $skip_type_clause = false, $skip_default_clause =
false, $skip_notnull_clause = false)  {
+
+     /// First of all, convert integers to numbers if defined
+         if ($this->integer_to_number) {
+             if ($xmldb_field->getType() == XMLDB_TYPE_INTEGER) {
+                 $xmldb_field->setType(XMLDB_TYPE_NUMBER);
+             }
+         }
+     /// Same for floats
+         if ($this->float_to_number) {
+             if ($xmldb_field->getType() == XMLDB_TYPE_FLOAT) {
+                 $xmldb_field->setType(XMLDB_TYPE_NUMBER);
+             }
+         }
```

```
+
+     /// The name
+         $field = $this->getEncQuoted($xmldb_field->getName());
+     /// The type and length only if we don't want to skip it
+         if (!$skip_type_clause) {
+         /// The type and length (if the field isn't enum)
+             if (!$xmldb_field->getEnum() || $this->enum_inline_code == false) {
+                 $field .= ' ' . $this->getTypeSQL($xmldb_field->getType(), $xmldb_field-
>getLength(), $xmldb_field->getDecimals());
+             } else {
+             /// call to custom function
+                 $field .= ' ' . $this->getEnumSQL($xmldb_field);
+             }
+         }
+     /// The unsigned if supported
+         if ($this->unsigned_allowed && ($xmldb_field->getType() == XMLDB_TYPE_INTEGER ||
+                                         $xmldb_field->getType() == XMLDB_TYPE_NUMBER ||
+                                         $xmldb_field->getType() == XMLDB_TYPE_FLOAT)) {
+             if ($xmldb_field->getUnsigned()) {
+                 $field .= ' unsigned';
+             }
+         }
+     /// Calculate the not null clause
+         $notnull = '';
+     /// Only if we don't want to skip it
+         if (!$skip_notnull_clause) {
+             if ($xmldb_field->getNotNull()) {
+                 $notnull = ' NOT NULL';
+             } else {
+                 if ($this->specify_nulls) {
+                     $notnull = ' NULL';
+                 }
+             }
+         }
+     /// Calculate the default clause
+         if (!$skip_default_clause) { //Only if we don't want to skip it
+             $default = $this->getDefaultClause($xmldb_field);
+         } else {
+             $default = '';
+         }
+     /// Based on default_after_null, set both clauses properly
+         if ($this->default_after_null) {
+             $field .= $notnull . $default;
+         } else {
+             $field .= $default . $notnull;
+         }
+     /// The sequence
+         if ($xmldb_field->getSequence()) {
+             if($xmldb_field->getLength()<=9 && $this->sequence_name_small) {
+                 $sequencename=$this->sequence_name_small;
+             } else {
+                 $sequencename=$this->sequence_name;
+             }
+             $field .= ' ' . $sequencename;
+             if ($this->sequence_only) {
+             /// We only want the field name and sequence name to be printed
+             /// so, calculate it and return
```

```
+                 return $this->getEncQuoted($xmldb_field->getName()) . ' ' . $sequencename;
+             }
+         }
+         return $field;
+     }
+
+     /**
+      * Given one correct XMLDBKey, returns its specs
+      *//*
+     function getKeySQL ($xmldb_table, $xmldb_key) {
+
+         $key = '';
+
+         switch ($xmldb_key->getType()) {
+             case XMLDB_KEY_PRIMARY:
+                 if ($this->primary_keys) {
+                     if ($this->primary_key_name !== null) {
+                         $key = $this->getEncQuoted($this->primary_key_name);
+                     } else {
+                         $key = $this->getNameForObject($xmldb_table->getName(), implode(',
', $xmldb_key->getFields()), 'pk');
+                     }
+                     $key .= ' PRIMARY KEY (' . implode(', ', $this-
>getEncQuoted($xmldb_key->getFields())) . ')';
+                 }
+                 break;
+             case XMLDB_KEY_UNIQUE:
+                 if ($this->unique_keys) {
+                     $key = $this->getNameForObject($xmldb_table->getName(), implode(', ',
$xmldb_key->getFields()), 'uk');
+                     $key .= ' UNIQUE (' . implode(', ', $this->getEncQuoted($xmldb_key-
>getFields())) . ')';
+                 }
+                 break;
+             case XMLDB_KEY_FOREIGN:
+             case XMLDB_KEY_FOREIGN_UNIQUE:
+                 if ($this->foreign_keys) {
+                     $key = $this->getNameForObject($xmldb_table->getName(), implode(', ',
$xmldb_key->getFields()), 'fk');
+                     $key .= ' FOREIGN KEY (' . implode(', ', $this-
>getEncQuoted($xmldb_key->getFields())) . ')';
+                     $key .= ' REFERENCES ' . $this->getEncQuoted($this->prefix .
$xmldb_key->getRefTable());
+                     $key .= ' (' . implode(', ', $this->getEncQuoted($xmldb_key-
>getRefFields())) . ')';
+                 }
+                 break;
+         }
+
+         return $key;
+     }
+
+     /**
+      * Give one XMLDBField, returns the correct "default value" for the current
configuration
+      *//*
+     function getDefaultValue ($xmldb_field) {
```

```
+
+         $default = null;
+
+         if ($xmldb_field->getDefault() !== NULL) {
+             if ($xmldb_field->getType() == XMLDB_TYPE_CHAR ||
+                 $xmldb_field->getType() == XMLDB_TYPE_TEXT) {
+                     $default = "'" . addslashes($xmldb_field->getDefault()) . "'";
+             } else {
+                 $default = $xmldb_field->getDefault();
+             }
+         } else {
+         /// We force default '' for not null char columns without proper default
+         /// some day this should be out!
+             if ($this->default_for_char !== NULL &&
+                 $xmldb_field->getType() == XMLDB_TYPE_CHAR &&
+                 $xmldb_field->getNotNull()) {
+                 $default = "'" . $this->default_for_char . "'";
+             } else {
+             /// If the DB requires to explicity define some clause to drop one default, do
it here
+             /// never applying defaults to TEXT and BINARY fields
+                 if ($this->drop_default_clause_required &&
+                     $xmldb_field->getType() != XMLDB_TYPE_TEXT &&
+                     $xmldb_field->getType() != XMLDB_TYPE_BINARY && !$xmldb_field-
>getNotNull()) {
+                     $default = $this->drop_default_clause;
+                 }
+             }
+         }
+         return $default;
+     }
+
+     /**
+      * Given one XMLDBField, returns the correct "default clause" for the current
configuration
+      *//*
+     function getDefaultClause ($xmldb_field) {
+
+         $defaultvalue = $this->getDefaultValue ($xmldb_field);
+
+         if ($defaultvalue !== null) {
+             return ' DEFAULT ' . $defaultvalue;
+         } else {
+             return null;
+         }
+     }
+
+     /**
+      * Given one correct XMLDBTable and the new name, returns the SQL statements
+      * to rename it (inside one array)
+      *//*
+
+     function getRenameTableSQL($xmldb_table, $newname) {
+
+         $results = array();  //Array where all the sentences will be stored
+
+         $newt = new XMLDBTable($newname); //Temporal table for name calculations
```

```
+
+         $rename = str_replace('OLDNAME', $this->getTableName($xmldb_table), $this-
>rename_table_sql);
+         $rename = str_replace('NEWNAME', $this->getTableName($newt), $rename);
+
+         $results[] = $rename;
+
+     /// Call to getRenameTableExtraSQL() if $rename_table_extra_code is enabled. It will
add sequence regeneration code.
+         if ($this->rename_table_extra_code) {
+             $extra_sentences = $this->getRenameTableExtraSQL($xmldb_table, $newname);
+             $results = array_merge($results, $extra_sentences);
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one correct XMLDBTable and the new name, returns the SQL statements
+      * to drop it (inside one array)
+      *//*
+     function getDropTableSQL($xmldb_table) {
+
+         $results = array();  //Array where all the sentences will be stored
+
+         $drop = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>drop_table_sql);
+
+         $results[] = $drop;
+
+     /// call to getDropTableExtraSQL() if $drop_table_extra_code is enabled. It will add
sequence/trigger drop code.
+         if ($this->drop_table_extra_code) {
+             $extra_sentences = $this->getDropTableExtraSQL($xmldb_table);
+             $results = array_merge($results, $extra_sentences);
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to add
the field to the table
+      *//*
+     function getAddFieldSQL($xmldb_table, $xmldb_field) {
+
+         $results = array();
+
+     /// Get the quoted name of the table and field
+         $tablename = $this->getTableName($xmldb_table);
+
+     /// Build the standard alter table add
+         $altertable = 'ALTER TABLE ' . $tablename . ' ADD ' .
+                       $this->getFieldSQL($xmldb_field, $this->alter_column_skip_type,
+                                                        $this-
>alter_column_skip_default,
+                                                        $this-
```

```
>alter_column_skip_notnull);
+     /// Add the after clause if necesary
+         if ($this->add_after_clause && $xmldb_field->getPrevious()) {
+             $altertable .= ' after ' . $this->getEncQuoted($xmldb_field->getPrevious());
+         }
+         $results[] = $altertable;
+
+     /// If the DB has extra enum code
+         if ($this->enum_extra_code) {
+         /// If it's enum add the extra code
+             if ($xmldb_field->getEnum()) {
+                 $results[] = 'ALTER TABLE ' . $tablename . ' ADD ' . $this-
>getEnumExtraSQL($xmldb_table, $xmldb_field);
+             }
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to drop
the field from the table
+      *//*
+     function getDropFieldSQL($xmldb_table, $xmldb_field) {
+
+         $results = array();
+
+     /// Get the quoted name of the table and field
+         $tablename = $this->getTableName($xmldb_table);
+         $fieldname = $this->getEncQuoted($xmldb_field->getName());
+
+     /// Build the standard alter table drop
+         $results[] = 'ALTER TABLE ' . $tablename . ' DROP COLUMN ' . $fieldname;
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to alter
the field in the table
+      *//*
+     function getAlterFieldSQL($xmldb_table, $xmldb_field) {
+
+         $results = array();
+
+     /// Always specify NULLs in alter fields because we can change not nulls to nulls
+         $this->specify_nulls = true;
+
+     /// Get the quoted name of the table and field
+         $tablename = $this->getTableName($xmldb_table);
+         $fieldname = $this->getEncQuoted($xmldb_field->getName());
+
+     /// Build de alter sentence using the alter_column_sql template
+         $alter = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>alter_column_sql);
+         $alter = str_replace('COLUMNSPECS', $this->getFieldSQL($xmldb_field, $this-
>alter_column_skip_type,
```

```
+                                                                       $this-
>alter_column_skip_default,
+                                                                       $this-
>alter_column_skip_notnull), $alter);
+
+     /// Add the after clause if necesary
+         if ($this->add_after_clause && $xmldb_field->getPrevious()) {
+             $alter .= ' after ' . $this->getEncQuoted($xmldb_field->getPrevious());
+         }
+
+     /// Build the standard alter table modify
+         $results[] = $alter;
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to modify
the enum of the field in the table
+      *//*
+     function getModifyEnumSQL($xmldb_table, $xmldb_field) {
+
+         $results = array();
+
+     /// Get the quoted name of the table and field
+         $tablename = $this->getTableName($xmldb_table);
+         $fieldname = $this->getEncQuoted($xmldb_field->getName());
+
+     /// Decide if we are going to create or to drop the enum (based exclusively in the
values passed!)
+         if (!$xmldb_field->getEnum()) {
+             $results = $this->getDropEnumSQL($xmldb_table, $xmldb_field); //Drop
+         } else {
+             $results = $this->getCreateEnumSQL($xmldb_table, $xmldb_field); //Create/modify
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to modify
the default of the field in the table
+      *//*
+     function getModifyDefaultSQL($xmldb_table, $xmldb_field) {
+
+         $results = array();
+
+     /// Get the quoted name of the table and field
+         $tablename = $this->getTableName($xmldb_table);
+         $fieldname = $this->getEncQuoted($xmldb_field->getName());
+
+     /// Decide if we are going to create/modify or to drop the default
+         if ($xmldb_field->getDefault() === null) {
+             $results = $this->getDropDefaultSQL($xmldb_table, $xmldb_field); //Drop
+         } else {
+             $results = $this->getCreateDefaultSQL($xmldb_table, $xmldb_field);
//Create/modify
```

```
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one correct XMLDBField and the new name, returns the SQL statements
+      * to rename it (inside one array)
+      *//*
+     function getRenameFieldSQL($xmldb_table, $xmldb_field, $newname) {
+
+         $results = array();  //Array where all the sentences will be stored
+
+     /// Although this is checked in ddllib - rename_field() - double check
+     /// that we aren't trying to rename one "id" field. Although it could be
+     /// implemented (if adding the necessary code to rename sequences, defaults,
+     /// triggers... and so on under each getRenameFieldExtraSQL() function, it's
+     /// better to forbide it, mainly because this field is the default PK and
+     /// in the future, a lot of FKs can be pointing here. So, this field, more
+     /// or less, must be considered inmutable!
+         if ($xmldb_field->getName() == 'id') {
+             return array();
+         }
+
+         $rename = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>rename_column_sql);
+         $rename = str_replace('OLDFIELDNAME', $this->getEncQuoted($xmldb_field->getName()),
$rename);
+         $rename = str_replace('NEWFIELDNAME', $this->getEncQuoted($newname), $rename);
+
+         $results[] = $rename;
+
+     /// Call to getRenameFieldExtraSQL() if $rename_column_extra_code is enabled (will add
some required sentences)
+         if ($this->rename_column_extra_code) {
+             $extra_sentences = $this->getRenameFieldExtraSQL($xmldb_table, $xmldb_field,
$newname);
+             $results = array_merge($results, $extra_sentences);
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBKey, return the SQL statements needded to add the
key to the table
+      * note that undelying indexes will be added as parametrised by $xxxx_keys and
$xxxx_index parameters
+      *//*
+     function getAddKeySQL($xmldb_table, $xmldb_key) {
+
+         $results = array();
+
+     /// Just use the CreateKeySQL function
+         if ($keyclause = $this->getKeySQL($xmldb_table, $xmldb_key)) {
+             $key = 'ALTER TABLE ' . $this->getTableName($xmldb_table) .
+                 ' ADD CONSTRAINT ' . $keyclause;
```

```
+            $results[] = $key;
+        }
+
+    /// If we aren't creating the keys OR if the key is XMLDB_KEY_FOREIGN (not underlying
index generated
+    /// automatically by the RDBMS) create the underlying (created by us) index (if doesn't
exists)
+        if (!$keyclause || $xmldb_key->getType() == XMLDB_KEY_FOREIGN) {
+        /// Only if they don't exist
+            if ($xmldb_key->getType() == XMLDB_KEY_FOREIGN) {  ///Calculate type of index
based on type ok key
+                $indextype = XMLDB_INDEX_NOTUNIQUE;
+            } else {
+                $indextype = XMLDB_INDEX_UNIQUE;
+            }
+            $xmldb_index = new XMLDBIndex('anyname');
+            $xmldb_index->setAttributes($indextype, $xmldb_key->getFields());
+            if (!index_exists($xmldb_table, $xmldb_index)) {
+                $results = array_merge($results, $this->getAddIndexSQL($xmldb_table,
$xmldb_index));
+            }
+        }
+
+    /// If the key is XMLDB_KEY_FOREIGN_UNIQUE, create it as UNIQUE too
+        if ($xmldb_key->getType() == XMLDB_KEY_FOREIGN_UNIQUE && $this->unique_keys) {
+        ///Duplicate the key
+            $xmldb_key->setType(XMLDB_KEY_UNIQUE);
+            $results = array_merge($results, $this->getAddKeySQL($xmldb_table,
$xmldb_key));
+        }
+
+    /// Return results
+        return $results;
+    }
+
+    /**
+     * Given one XMLDBTable and one XMLDBIndex, return the SQL statements needded to drop
the index from the table
+     *//*
+    function getDropKeySQL($xmldb_table, $xmldb_key) {
+
+        $results = array();
+
+    /// Get the key name (note that this doesn't introspect DB, so could cause some
problems sometimes!)
+    /// TODO: We'll need to overwrite the whole getDropKeySQL() method inside each DB to do
the proper queries
+    /// against the dictionary or require ADOdb to support it or change the find_key_name()
method to
+    /// perform DB introspection directly. But, for now, as we aren't going to enable
referential integrity
+    /// it won't be a problem at all
+        $dbkeyname = find_key_name($xmldb_table, $xmldb_key);
+
+    /// Only if such type of key generation is enabled
+        $dropkey = false;
+        switch ($xmldb_key->getType()) {
```

```
+             case XMLDB_KEY_PRIMARY:
+                 if ($this->primary_keys) {
+                     $template = $this->drop_primary_key;
+                     $dropkey = true;
+                 }
+                 break;
+             case XMLDB_KEY_UNIQUE:
+                 if ($this->unique_keys) {
+                     $template = $this->drop_unique_key;
+                     $dropkey = true;
+                 }
+                 break;
+             case XMLDB_KEY_FOREIGN_UNIQUE:
+             case XMLDB_KEY_FOREIGN:
+                 if ($this->foreign_keys) {
+                     $template = $this->drop_foreign_key;
+                     $dropkey = true;
+                 }
+                 break;
+         }
+     /// If we have decided to drop the key, let's do it
+         if ($dropkey) {
+         /// Replace TABLENAME, CONSTRAINTTYPE and KEYNAME as needed
+             $dropsql = str_replace('TABLENAME', $this->getTableName($xmldb_table),
$template);
+             $dropsql = str_replace('KEYNAME', $dbkeyname, $dropsql);
+
+             $results[] = $dropsql;
+         }
+
+     /// If we aren't dropping the keys OR if the key is XMLDB_KEY_FOREIGN (not underlying
index generated
+     /// automatically by the RDBMS) drop the underlying (created by us) index (if exists)
+         if (!$dropkey || $xmldb_key->getType() == XMLDB_KEY_FOREIGN) {
+         /// Only if they exist
+             $xmldb_index = new XMLDBIndex('anyname');
+             $xmldb_index->setAttributes(XMLDB_INDEX_UNIQUE, $xmldb_key->getFields());
+             if (index_exists($xmldb_table, $xmldb_index)) {
+                 $results = array_merge($results, $this->getDropIndexSQL($xmldb_table,
$xmldb_index));
+             }
+         }
+
+     /// If the key is XMLDB_KEY_FOREIGN_UNIQUE, drop the UNIQUE too
+         if ($xmldb_key->getType() == XMLDB_KEY_FOREIGN_UNIQUE && $this->unique_keys) {
+         ///Duplicate the key
+             $xmldb_key->setType(XMLDB_KEY_UNIQUE);
+             $results = array_merge($results, $this->getDropKeySQL($xmldb_table,
$xmldb_key));
+         }
+
+     /// Return results
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBKey, return the SQL statements needded to rename
```

```
the key in the table
+      * Experimental! Shouldn't be used at all!
+      *//*
+     function getRenameKeySQL($xmldb_table, $xmldb_key, $newname) {
+
+         $results = array();
+
+     /// Get the real key name
+         $dbkeyname = find_key_name($xmldb_table, $xmldb_key);
+
+     /// Check we are really generating this type of keys
+         if (($xmldb_key->getType() == XMLDB_KEY_PRIMARY && !$this->primary_keys) ||
+             ($xmldb_key->getType() == XMLDB_KEY_UNIQUE && !$this->unique_keys) ||
+             ($xmldb_key->getType() == XMLDB_KEY_FOREIGN && !$this->foreign_keys) ||
+             ($xmldb_key->getType() == XMLDB_KEY_FOREIGN_UNIQUE && !$this->unique_keys && !
$this->foreign_keys)) {
+         /// We aren't generating this type of keys, delegate to child indexes
+             $xmldb_index = new XMLDBIndex($xmldb_key->getName());
+             $xmldb_index->setFields($xmldb_key->getFields());
+             return $this->getRenameIndexSQL($xmldb_table, $xmldb_index, $newname);
+         }
+
+     /// Arrived here so we are working with keys, lets rename them
+     /// Replace TABLENAME and KEYNAME as needed
+         $renamesql = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>rename_key_sql);
+         $renamesql = str_replace('OLDKEYNAME', $dbkeyname, $renamesql);
+         $renamesql = str_replace('NEWKEYNAME', $newname, $renamesql);
+
+     /// Some DB doesn't support key renaming so this can be empty
+         if ($renamesql) {
+             $results[] = $renamesql;
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBIndex, return the SQL statements needded to add
the index to the table
+      *//*
+     function getAddIndexSQL($xmldb_table, $xmldb_index) {
+
+     /// Just use the CreateIndexSQL function
+         return $this->getCreateIndexSQL($xmldb_table, $xmldb_index);
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBIndex, return the SQL statements needded to drop
the index from the table
+      *//*
+     function getDropIndexSQL($xmldb_table, $xmldb_index) {
+
+         $results = array();
+
+     /// Get the real index name
+         $dbindexname = find_index_name($xmldb_table, $xmldb_index);
```

```
+
+     /// Replace TABLENAME and INDEXNAME as needed
+         $dropsql = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>drop_index_sql);
+         $dropsql = str_replace('INDEXNAME', $dbindexname, $dropsql);
+
+         $results[] = $dropsql;
+
+         return $results;
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBIndex, return the SQL statements needded to rename
the index in the table
+      * Experimental! Shouldn't be used at all!
+      *//*
+     function getRenameIndexSQL($xmldb_table, $xmldb_index, $newname) {
+
+         $results = array();
+
+     /// Get the real index name
+         $dbindexname = find_index_name($xmldb_table, $xmldb_index);
+
+     /// Replace TABLENAME and INDEXNAME as needed
+         $renamesql = str_replace('TABLENAME', $this->getTableName($xmldb_table), $this-
>rename_index_sql);
+         $renamesql = str_replace('OLDINDEXNAME', $dbindexname, $renamesql);
+         $renamesql = str_replace('NEWINDEXNAME', $newname, $renamesql);
+
+     /// Some DB doesn't support index renaming (MySQL) so this can be empty
+         if ($renamesql) {
+             $results[] = $renamesql;
+         }
+
+         return $results;
+     }
+
+     /**
+      * Given three strings (table name, list of fields (comma separated) and suffix),
+      * create the proper object name quoting it if necessary.
+      *
+      * IMPORTANT: This function must be used to CALCULATE NAMES of objects TO BE CREATED,
+      *            NEVER TO GUESS NAMES of EXISTING objects!!!
+      *//*
+     function getNameForObject($tablename, $fields, $suffix='') {
+
+         $name = '';
+
+     /// Implement one basic cache to avoid object name duplication
+     /// and to speed up repeated queries for the same objects
+         if (!isset($used_names)) {
+             static $used_names = array();
+         }
+
+     /// If this exact object has been requested, return it
+         if (array_key_exists($tablename.'-'.$fields.'-'.$suffix, $used_names)) {
+             return $used_names[$tablename.'-'.$fields.'-'.$suffix];
```

```
+         }
+
+     /// Use standard naming. See http://docs.moodle.org/en/XMLDB_key_and_index_naming
+         $tablearr = explode ('_', $tablename);
+         foreach ($tablearr as $table) {
+             $name .= substr(trim($table),0,4);
+         }
+         $name .= '_';
+         $fieldsarr = explode (',', $fields);
+         foreach ($fieldsarr as $field) {
+             $name .= substr(trim($field),0,3);
+         }
+     /// Prepend the prefix
+         $name = $this->prefix . $name;
+
+         $name = substr(trim($name), 0, $this->names_max_length - 1 - strlen($suffix));
//Max names_max_length
+
+     /// Add the suffix
+         $namewithsuffix = $name;
+         if ($suffix) {
+             $namewithsuffix = $namewithsuffix . '_' . $suffix;
+         }
+
+     /// If the calculated name is in the cache, or if we detect it by introspecting the DB
let's modify if
+         if (in_array($namewithsuffix, $used_names) || $this->isNameInUse($namewithsuffix,
$suffix, $tablename)) {
+             $counter = 2;
+         /// If have free space, we add 2
+             if (strlen($namewithsuffix) < $this->names_max_length) {
+                 $newname = $name . $counter;
+         /// Else replace the last char by 2
+             } else {
+                 $newname = substr($name, 0, strlen($name)-1) . $counter;
+             }
+             $newnamewithsuffix = $newname;
+             if ($suffix) {
+                 $newnamewithsuffix = $newnamewithsuffix . '_' . $suffix;
+             }
+         /// Now iterate until not used name is found, incrementing the counter
+             while (in_array($newnamewithsuffix, $used_names) || $this-
>isNameInUse($newnamewithsuffix, $suffix, $tablename)) {
+                 $counter++;
+                 $newname = substr($name, 0, strlen($newname)-1) . $counter;
+                 $newnamewithsuffix = $newname;
+                 if ($suffix) {
+                     $newnamewithsuffix = $newnamewithsuffix . '_' . $suffix;
+                 }
+             }
+             $namewithsuffix = $newnamewithsuffix;
+         }
+
+     /// Add the name to the cache
+         $used_names[$tablename.'-'.$fields.'-'.$suffix] = $namewithsuffix;
+
+     /// Quote it if necessary (reserved words)
```

```
+         $namewithsuffix = $this->getEncQuoted($namewithsuffix);
+
+         return $namewithsuffix;
+     }
+
+     /**
+      * Given any string (or one array), enclose it by the proper quotes
+      * if it's a reserved word
+      *//*
+     function getEncQuoted($input) {
+
+         if (is_array($input)) {
+             foreach ($input as $key=>$content) {
+                 $input[$key] = $this->getEncQuoted($content);
+             }
+             return $input;
+         } else {
+         /// Always lowercase
+             $input = strtolower($input);
+         /// if reserved or quote_all, quote it
+             if ($this->quote_all || in_array($input, $this->reserved_words)) {
+                 $input = $this->quote_string . $input . $this->quote_string;
+             }
+             return $input;
+         }
+     }
+
+     /**
+      * Given one XMLDB Statement, build the needed SQL insert sentences to execute it
+      *//*
+     function getExecuteInsertSQL($statement) {
+
+          $results = array();  //Array where all the sentences will be stored
+
+          if ($sentences = $statement->getSentences()) {
+              foreach ($sentences as $sentence) {
+              /// Get the list of fields
+                  $fields = $statement->getFieldsFromInsertSentence($sentence);
+              /// Get the values of fields
+                  $values = $statement->getValuesFromInsertSentence($sentence);
+              /// Look if we have some CONCAT value and transform it dinamically
+                  foreach($values as $key => $value) {
+                  /// Trim single quotes
+                      $value = trim($value,"'");
+                      if (stristr($value, 'CONCAT') !== false){
+                      /// Look for data between parentesis
+                          preg_match("/CONCAT\s*\((.*)\)$/is", trim($value), $matches);
+                          if (isset($matches[1])) {
+                              $part = $matches[1];
+                          /// Convert the comma separated string to an array
+                              $arr = XMLDBObject::comma2array($part);
+                              if ($arr) {
+                                  $value = $this->getConcatSQL($arr);
+                              }
+                          }
+                      }
+                  /// Values to be sent to DB must be properly escaped
```

```
+                    $value = addslashes($value);
+                /// Back trimmed quotes
+                    $value = "'" . $value . "'";
+                /// Back to the array
+                    $values[$key] = $value;
+                }
+
+             /// Iterate over fields, escaping them if necessary
+                foreach($fields as $key => $field) {
+                    $fields[$key] = $this->getEncQuoted($field);
+                }
+             /// Build the final SQL sentence and add it to the array of results
+             $sql = 'INSERT INTO ' . $this->getEncQuoted($this->prefix . $statement-
>getTable()) .
+                        '(' . implode(', ', $fields) . ') ' .
+                        'VALUES (' . implode(', ', $values) . ')';
+                $results[] = $sql;
+             }
+
+          }
+          return $results;
+     }
+
+     /**
+      * Given one array of elements, build de proper CONCAT expresion, based
+      * in the $concat_character setting. If such setting is empty, then
+      * MySQL's CONCAT function will be used instead
+      *//*
+     function getConcatSQL($elements) {
+
+     /// Replace double quoted elements by single quotes
+         foreach($elements as $key => $element) {
+             $element = trim($element);
+             if (substr($element, 0, 1) == '"' &&
+                 substr($element, -1, 1) == '"') {
+                     $elements[$key] = "'" . trim($element, '"') . "'";
+             }
+         }
+
+     /// Now call the standard sql_concat() DML function
+         return call_user_func_array('sql_concat', $elements);
+     }
+
+     /**
+      * Given one string (or one array), ends it with statement_end
+      *//*
+     function getEndedStatements ($input) {
+
+         if (is_array($input)) {
+             foreach ($input as $key=>$content) {
+                 $input[$key] = $this->getEndedStatements($content);
+             }
+             return $input;
+         } else {
+             $input = trim($input) . $this->statement_end;
+             return $input;
+         }
```

```
+     }
+
+     /**
+      * Returns the name (string) of the sequence used in the table for the autonumeric pk
+      * Only some DB have this implemented
+      *//*
+     function getSequenceFromDB($xmldb_table) {
+         return false;
+     }
+
+     /**
+      * Given one object name and it's type (pk, uk, fk, ck, ix, uix, seq, trg)
+      * return if such name is currently in use (true) or no (false)
+      * (MySQL requires the whole XMLDBTable object to be specified, so we add it always)
+      * (invoked from getNameForObject()
+      * Only some DB have this implemented
+      *//*
+     function isNameInUse($object_name, $type, $table_name) {
+         return false; //For generators not implementing introspecion,
+                       //we always return with the name being free to be used
+     }
+
+
+ /// ALL THESE FUNCTION MUST BE CUSTOMISED BY ALL THE XMLDGenerator classes
+
+     /**
+      * Given one XMLDB Type, lenght and decimals, returns the DB proper SQL type
+      */
+     function getTypeSQL ($xmldb_type, $xmldb_length=null, $xmldb_decimals=null) {
+         switch ($xmldb_type) {
+             case XMLDB_TYPE_INTEGER:
+                 //Integer
+                 if (empty($xmldb_length)) {
+                     $xmldb_length = 10;
+                 }
+                 if ($xmldb_length > 9) {
+                     $dbtype = 'BIGINT';
+                 } else if ($xmldb_length > 4) {
+                     $dbtype = 'INTEGER';
+                 } else {
+                     $dbtype = 'SMALLINT';
+                 }
+                 break;
+             case XMLDB_TYPE_NUMBER:
+                 //Decimal number
+                 /// 31 is the max allowed
+                 if ($xmldb_length > 31) {
+                     $xmldb_length = 31;
+                 }
+                 $dbtype = 'DECIMAL';
+                 if (!empty($xmldb_length)) {
+                     $dbtype .= '(' . $xmldb_length;
+                     if (!empty($xmldb_decimals)) {
+                         $dbtype .= ',' . $xmldb_decimals;
+                     }
+                     $dbtype .= ')';
+                 }
```

```
+                 break;
+             case XMLDB_TYPE_FLOAT:
+                 //Floating Point number
+                 $dbtype = 'DOUBLE';
+                 if (!empty($xmldb_decimals)) {
+                     if ($xmldb_decimals < 6) {
+                         $dbtype = 'FLOAT';
+                     }
+                 }
+                 break;
+             case XMLDB_TYPE_CHAR:
+                 //String
+                 $dbtype = 'VARCHAR';
+                 if (empty($xmldb_length)) {
+                     $xmldb_length='255';
+                 }
+                 $dbtype .= '(' . $xmldb_length . ')';
+                 break;
+             case XMLDB_TYPE_TEXT:
+                 //Text
+                 $dbtype = 'CLOB';
+                 break;
+             case XMLDB_TYPE_BINARY:
+                 //Binary
+                 $dbtype = 'BLOB';
+                 break;
+             case XMLDB_TYPE_DATETIME:
+                 //Datetime
+                 $dbtype = 'TIMESTAMP';
+                 break;
+             case XMLDB_TYPE_TIMESTAMP:
+                 //Timestamp
+                 $dbtype = 'TIMESTAMP';
+                 break;
+         }
+         return $dbtype;
+     }
+
+     /**
+      * Given one XMLDB Field, return its enum SQL to be added inline with the column
definition
+      *//*
+     function getEnumSQL ($xmldb_field) {
+         return 'code for inline enum declaration goes to function getEnumSQL(). Can be
disabled with enum_inline_code=false';
+     }
+
+     /**
+      * Returns the code needed to create one enum for the xmldb_table and xmldb_field
passes
+      */
+     function getEnumExtraSQL ($xmldb_table, $xmldb_field) {
+       $sql = 'CONSTRAINT ' . $this->getNameForObject($xmldb_table->getName(), $xmldb_field-
>getName(), 'ck');
+         $sql.= ' CHECK (' . $this->getEncQuoted($xmldb_field->getName()) . ' IN (' .
implode(', ', $xmldb_field->getEnumValues()) . '))';
+         return $sql;
```

```
+     }
+
+     /**
+      * Returns the code (array of statements) needed to execute extra statements on field
rename
+      *//*
+     function getRenameFieldExtraSQL ($xmldb_table, $xmldb_field) {
+         return array('Code for field rename goes to getRenameFieldExtraSQL(). Can be
disabled with rename_column_extra_code=false;');
+     }
+
+     /**
+      * Returns the code (array of statements) needed
+      * to create one sequence for the xmldb_table and xmldb_field passes
+      *//*
+     function getCreateSequenceSQL ($xmldb_table, $xmldb_field) {
+         return array('Code for extra sequence SQL goes to getCreateSequenceSQL(). Can be
disabled with sequence_extra_code=false');
+     }
+
+     /**
+      * Returns the code (array of statements) needed to add one comment to the table
+      *//*
+     function getCommentSQL ($xmldb_table) {
+         return array('Code for table comment goes to getCommentSQL(). Can be disabled with
add_table_comments=false;');
+     }
+
+     /**
+      * Returns the code (array of statements) needed to execute extra statements on table
rename
+      *//*
+     function getRenameTableExtraSQL ($xmldb_table) {
+         return array('Code for table rename goes to getRenameTableExtraSQL(). Can be
disabled with rename_table_extra_code=false;');
+     }
+
+     /**
+      * Returns the code (array of statements) needed to execute extra statements on table
drop
+      *//*
+     function getDropTableExtraSQL ($xmldb_table) {
+         return array('Code for table drop goes to getDropTableExtraSQL(). Can be disabled
with drop_table_extra_code=false;');
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to drop
its enum
+      * (usually invoked from getModifyEnumSQL()
+      */
+     function getDropEnumSQL($xmldb_table, $xmldb_field) {
+     /// Let's introspect to know the real name of the check constraint
+         if ($check_constraints = $this->getCheckConstraintsFromDB($xmldb_table,
$xmldb_field)) {
+             $check_constraint = array_shift($check_constraints); /// Get the 1st (should be
only one)
```

```
+            $constraint_name = strtolower($check_constraint->name); /// Extract the REAL
name
+         /// All we have to do is to drop the check constraint
+             return array('ALTER TABLE ' . $this->getTableName($xmldb_table) .
+                      ' DROP CONSTRAINT ' . $constraint_name);
+         } else { /// Constraint not found. Nothing to do
+             return array();
+         }
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to add
its enum
+      * (usually invoked from getModifyEnumSQL()
+      */
+     function getCreateEnumSQL($xmldb_table, $xmldb_field) {
+     /// All we have to do is to create the check constraint
+         return array('ALTER TABLE ' . $this->getTableName($xmldb_table) .
+                      ' ADD ' . $this->getEnumExtraSQL($xmldb_table, $xmldb_field));
+     }
+
+     /**
+      * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to drop
its default
+      * (usually invoked from getModifyDefaultSQL()
+      */
+     function getDropDefaultSQL($xmldb_table, $xmldb_field) {
+     /// Just a wrapper over the getAlterFieldSQL() function for DB2 that
+     /// is capable of handling defaults
+         return $this->getAlterFieldSQL($xmldb_table, $xmldb_field);
+     }
+
+     /**
+      * Given one XMLDBTable and one optional XMLDBField, return one array with all the
check
+      * constrainst found for that table (or field). Must exist for each DB supported.
+      * (usually invoked from find_check_constraint_name)
+      */
+     function getCheckConstraintsFromDB($xmldb_table, $xmldb_field=null) {
+         $results = array();
+
+       $tablename = strtoupper($this->getTableName($xmldb_table));
+       /// Source:
http://www.ibm.com/developerworks/data/library/techarticle/dm-0401melnyk/index.html
+         if ($xmldb_field == null) {
+             if ($constraints = get_records_sql("SELECT c.CONSTNAME AS name
+                                                 FROM SYSCAT.CHECKS AS c
+                                                 WHERE c.TABNAME = '{$tablename}'
+                                                 AND c.TYPE = 'C'
+                                                 AND c.CONSTNAME NOT LIKE 'SYS%'")) {
+                 foreach ($constraints as $constraint) {
+                     $results[$constraint->name] = $constraint;
+                 }
+             }
+         }
+         else {
+             $colname = strtoupper($xmldb_field->getName());
```

```
+            if ($constraints = get_records_sql("SELECT c.CONSTNAME AS name
+                                                FROM SYSCAT.CHECKS AS c
+                                                WHERE c.TABNAME = '{$tablename}'
+                                                AND c.COLNAME = '{$colname}'
+                                                AND c.TYPE = 'C'
+                                                AND c.CONSTNAME NOT LIKE 'SYS%'")) {
+                foreach ($constraints as $constraint) {
+                    $results[$constraint->name] = $constraint;
+                }
+            }
+        }
+        return $results();
+    }
+
+    /**
+     * Given one XMLDBTable and one XMLDBField, return the SQL statements needded to add
its default
+     * (usually invoked from getModifyDefaultSQL()
+     */
+    function getCreateDefaultSQL($xmldb_table, $xmldb_field) {
+    /// Just a wrapper over the getAlterFieldSQL() function for DB2 that
+    /// is capable of handling defaults
+        return $this->getAlterFieldSQL($xmldb_table, $xmldb_field);
+    }
+
+    /**
+     * Returns an array of reserved words (lowercase) for this DB
+     * You MUST provide the real list for each DB inside every XMLDB class
+     */
+    function getReservedWords() {
+    /// DB2 reserved words:
ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2s1e80.pdf  Appendex G
+        $reserved_words = array (
+            'absolute', 'action', 'add', 'admin', 'after', 'aggregate', 'alias', 'all',
'allocate',
+            'allow', 'alter', 'and', 'any', 'application', 'are', 'array', 'as', 'asc',
'assertion',
+            'associate', 'asutime', 'at', 'audit', 'authorization', 'aux', 'auxiliary',
'before',
+            'begin', 'between', 'binary', 'bit', 'blob', 'boolean', 'both', 'breadth',
'bufferpool',
+            'by', 'cache', 'call', 'called', 'capture', 'cardinality', 'cascade',
'cascaded', 'case',
+            'cast', 'catalog', 'ccsid', 'char', 'character', 'check', 'class', 'clob',
'close',
+            'cluster', 'collate', 'collation', 'collection', 'collid', 'column', 'comment',
'commit',
+            'completion', 'concat', 'condition', 'connect', 'connection', 'constraint',
'constraints',
+            'constructor', 'contains', 'continue', 'corresponding', 'count', 'count_big',
'create',
+            'cross', 'cube', 'current', 'current_date', 'current_lc_ctype', 'current_path',
+            'current_role', 'current_server', 'current_time', 'current_timestamp',
'current_timezone',
+            'current_user', 'cursor', 'cycle', 'data', 'database', 'date', 'day', 'days',
'db2general',
+            'db2genrl', 'db2sql', 'dbinfo', 'deallocate', 'dec', 'decimal', 'declare',
```

```
'default',
+             'defaults', 'deferrable', 'deferred', 'definition', 'delete', 'depth', 'deref',
'desc',
+             'describe', 'descriptor', 'destroy', 'destructor', 'deterministic',
'diagnostics',
+             'dictionary', 'disallow', 'disconnect', 'distinct', 'do', 'domain', 'double',
'drop',
+             'dsnhattr', 'dssize', 'dynamic', 'each', 'editproc', 'else', 'elseif',
'encoding', 'end',
+             'end-exec', 'end-exec1', 'equals', 'erase', 'escape', 'every', 'except',
'exception',
+             'excluding', 'exec', 'execute', 'exists', 'exit', 'external', 'false',
'fenced', 'fetch',
+             'fieldproc', 'file', 'final', 'first', 'float', 'for', 'foreign', 'found',
'free', 'from',
+             'full', 'function', 'general', 'generated', 'get', 'global', 'go', 'goto',
'grant',
+             'graphic', 'group', 'grouping', 'handler', 'having', 'hold', 'host', 'hour',
'hours',
+             'identity', 'if', 'ignore', 'immediate', 'in', 'including', 'increment',
'index',
+             'indicator', 'inherit', 'initialize', 'initially', 'inner', 'inout', 'input',
'insensitive',
+             'insert', 'int', 'integer', 'integrity', 'intersect', 'interval', 'into', 'is',
'isobid',
+             'isolation', 'iterate', 'jar', 'java', 'join', 'key', 'label', 'language',
'large', 'last',
+             'lateral', 'lc_ctype', 'leading', 'leave', 'left', 'less', 'level', 'like',
'limit',
+             'linktype', 'local', 'locale', 'localtime', 'localtimestamp', 'locator',
'locators', 'lock',
+             'lockmax', 'locksize', 'long', 'loop', 'map', 'match', 'maxvalue',
'microsecond',
+             'microseconds', 'minute', 'minutes', 'minvalue', 'mode', 'modifies', 'modify',
'module',
+             'month', 'months', 'names', 'national', 'natural', 'nchar', 'nclob', 'new',
'new_table',
+             'next', 'no', 'nocache', 'nocycle', 'nodename', 'nodenumber', 'nomaxvalue',
'nominvalue',
+             'none', 'noorder', 'not', 'null', 'nulls', 'numeric', 'numparts', 'obid',
'object', 'of',
+             'off', 'old', 'old_table', 'on', 'only', 'open', 'operation', 'optimization',
'optimize',
+             'option', 'or', 'order', 'ordinality', 'out', 'outer', 'output', 'overriding',
'package',
+             'pad', 'parameter', 'parameters', 'part', 'partial', 'partition', 'path',
'piecesize', 'plan',
+             'position', 'postfix', 'precision', 'prefix', 'preorder', 'prepare',
'preserve', 'primary',
+             'prior', 'priqty', 'privileges', 'procedure', 'program', 'psid', 'public',
'queryno', 'read',
+             'reads', 'real', 'recovery', 'recursive', 'ref', 'references', 'referencing',
'relative',
+             'release', 'rename', 'repeat', 'reset', 'resignal', 'restart', 'restrict',
'result',
+             'result_set_locator', 'return', 'returns', 'revoke', 'right', 'role',
'rollback', 'rollup',
```

```
+             'routine', 'row', 'rows', 'rrn', 'run', 'savepoint', 'schema', 'scope',
'scratchpad', 'scroll',
+             'search', 'second', 'seconds', 'secqty', 'section', 'security', 'select',
'sensitive',
+             'sequence', 'session', 'session_user', 'set', 'sets', 'signal', 'simple',
'size', 'smallint',
+             'some', 'source', 'space', 'specific', 'specifictype', 'sql', 'sqlexception',
'sqlid',
+             'sqlstate', 'sqlwarning', 'standard', 'start', 'state', 'statement', 'static',
'stay',
+             'stogroup', 'stores', 'structure', 'style', 'subpages', 'substring', 'synonym',
'sysfun',
+             'sysibm', 'sysproc', 'system', 'system_user', 'table', 'tablespace',
'temporary', 'terminate',
+             'than', 'then', 'time', 'timestamp', 'timezone_hour', 'timezone_minute', 'to',
'trailing',
+             'transaction', 'translation', 'treat', 'trigger', 'trim', 'true', 'type',
'under', 'undo',
+             'union', 'unique', 'unknown', 'unnest', 'until', 'update', 'usage', 'user',
'using', 'validproc',
+             'value', 'values', 'varchar', 'variable', 'variant', 'varying', 'vcat', 'view',
'volumes',
+             'when', 'whenever', 'where', 'while', 'with', 'without', 'wlm', 'work',
'write', 'year', 'years',
+             'zone'
+         );
+         return $reserved_words;
+     }
+
+     /**
+      * Returns an array of tables to be built without prefix (lowercase)
+      * It's enough to keep updated here this function.
+      *//*
+     function getTablesWithoutPrefix() {
+     /// Some well-known tables to be created without prefix
+         $tables = array (
+             'adodb_logsql'
+         );
+         return $tables;
+     }*/
+ }
+
+ ?>
Binary files moodle-1.9.6/moodle/pix/f/edit.gif and moodle-db2-tip/moodle/pix/f/edit.gif
differ
diff -crBN moodle-1.9.6/moodle/search/indexlib.php moodle-db2-tip/moodle/search/indexlib.php
*** moodle-1.9.6/moodle/search/indexlib.php 2009-07-22 17:05:33.000000000 -0700
--- moodle-db2-tip/moodle/search/indexlib.php   2009-12-11 16:54:18.000000000 -0800
***************
*** 68,76 ****
          $admin_tables = $db->MetaTables();

          //TODO: use new IndexDBControl class for database checks?
!
          //check if our search table exists
!         if (in_array($CFG->prefix.SEARCH_DATABASE_TABLE, $admin_tables)) {
              //retrieve database information if it does
```

```
                $db_exists = true;

--- 68,80 ----
            $admin_tables = $db->MetaTables();

            //TODO: use new IndexDBControl class for database checks?
!           if ($CFG->dbfamily == 'db2') {
!               $table_name = strtoupper($CFG->prefix.SEARCH_DATABASE_TABLE);
!           } else {
!               $table_name = $CFG->prefix.SEARCH_DATABASE_TABLE;
!           }
            //check if our search table exists
!           if (in_array($table_name, $admin_tables)) {
                //retrieve database information if it does
                $db_exists = true;

***************
*** 185,191 ****

            $table = SEARCH_DATABASE_TABLE;
            $tables = $db->MetaTables();
!           if (in_array($CFG->prefix.$table, $tables)) {
                return true;
            }
            else {
--- 189,200 ----

            $table = SEARCH_DATABASE_TABLE;
            $tables = $db->MetaTables();
!           if ($CFG->dbfamily == 'db2') {
!               $table_name = strtoupper($CFG->prefix.$table);
!           } else {
!               $table_name = $CFG->prefix.$table;
!           }
!           if (in_array($table_name, $tables)) {
                return true;
            }
            else {
```