Moodle Database Enrollment Groups and Grouping Support

External tables

In addition to tables required for course and user sync, the enhancement to support group enrolment will also require...

Grouping table
  Must contain at least a column for groupingidnumber and course
  Groupingname defaults to groupingidnumber if no grouping name column is supplied or the value in this is empty
  # course is required here because the same groupidnumber can exist on two different courses

Group table
  Must contain at least a column for groupidnumber and course
  Groupname defaults to groupidnumber if no group name column is supplied or the value in this is empty
  # course is required here because the same groupidnumber can exist on two different courses

Group membership table
  Must contain at least a column for user and groupidnumber and course
  # course is required here because the same groupidnumber could exist on two different courses

Assumptions

This code assumes that courses have been synced first using the existing course-syncing functionality

This code will not edit or delete groups or groupings which do not have an idnumber. This means that any groups or grouping created in the normal way (i.e. through the web interface) will not be affected.
Syncing of groups, groupings and group memberships are each optional - each can be enabled without either of the others. In practice, syncing group memberships will usually be associated with syncing groups.

Changes required to code

enrol/database/settings.php

- remotegroupingtable
- remotegroupingidnumberfield
- remotegroupcoursefield
- remotegroupingnamefield (optional - defaults to remotegroupingidnumberfield if not specified)
- remotegroupingdescriptionfield (optional - if not specified, or no value present, then grouping description is left blank)

- remotegrouptable
- remotegroupidnumberfield
- remotegroupcoursefield
- remotegroupnamefield (optional - defaults to remotegroupnamefield if not specified)
- remotegroupingfield (optional - if not specified, or no value present, then group will not be added to a grouping)
- localgroupingfield (used to identify grouping when importing groups in groupings - can be either id, idnumber, name)

- remotegroupmembershiptable
- remotegroupmembershipuserfield
- remotegroupmembershipgroupfield
- remotegroupmembershipcoursefield - only required when localgroupfield is set to "name" or "idnumber"
- localgroupfield (used to identify group when importing group membership - can be either id, idnumber, name) - when using name, or idnumber then the course field must be set
Sync Groupings

# this will be in its own method call which will be called before syncing groupings

if { remotegroupmembershiptable and remotegroupingidnumberfield } {

    Get a list (in memory) of all existing (internal) groupings which have an idNumber
    This list will be keyed on combination of the value of what is in remotegroupingcoursefield and remotegroupingidnumberfield
    Data stored in this list for each existing grouping will be:
    ● groupingname
    ● md5sum of groupingdescription (just to reduce memory footprint)

    Query the external groupings table to pull out all groupings
    Iterate across groupings from the external table
    Query database to check that the course referenced in the external grouping table actually exists in the internal course table
    If course can't be found then...
        Generate a warning: "Grouping "$groupingName" was not imported because it belongs to a non-existant course"
        Continue

    # Create any new grouping
    If external grouping doesn't already exist (this is checked by looking up the combined remotegroupingcoursefield and remotegroupingidnumberfield in the in-memory list of groupings) then...
    Make sure the grouping name doesn't clash with an existing grouping (without an idnumber)
    If it does generate a warning: "Grouping "$groupingName" was not imported because an existing grouping exists with the same name but no idNumber"
    Continue

    Create new grouping
Use value from remotegroupingidnumberfield for idNumber
Use value from remotegroupingnamefield (if set) for name - otherwise use value from remotegroupingidnumberfield
Use value from remotegroupingdescriptionfield (if set) for description - otherwise leave description blank
Set the courseId to the current courseId

# Edit existing groupings that have changed
If external grouping does currently exist then
  If grouping name from external table doesn't match current grouping name then
    Update name
  If groupingdescriptionfield is set and grouping description from external table doesn't match current grouping description then
    Update description
Delete this entry from in-memory list of internal groupings

# Delete any previously imported groupings which are no longer in external db
Iterate across all records remaining in the in-memory list of groupings
  See if there are any non-imported (i.e. without an idnumber) groups in this grouping
  If there are non-imported groups then...
    DO NOT DELETE THE GROUPING - the fact that the user has appropriated this grouping for some other purpose means we should keep it alive
  Otherwise...
    Delete the grouping membership records (i.e. details of which groups were in this grouping)
    Delete the grouping
    # Don't do anything to groups that were in this grouping
    # It is valid for these groups to exist without a grouping
    # If the groups don't exist any more then they won't be in the groups table and will be deleted when we sync that

Sync Groups

# this will be in its own method call which will be called before syncing group membership
if ( remotegrouptable and remotegroupidnumberfield ) {
  Get a list (in memory) of all existing (internal) groups which have an idNumber
  This list will be keyed on combination of the value in remotegroupcoursefield and remotegroupidnumberfield
  Data stored in this list for each existing grouping will be:
    groupname
md5sum of groupdescription (just to reduce memory footprint)

Query the external group table to pull out all groups
Iterate across groups from the external table

Query database to get the courseID and check that the course referenced in the external group table actually exists
If it doesn't exist generate a warning: "Group "$groupName" was not imported because it belongs to a non-existant course"
Continue

# Create any new group
If external group doesn't already exist (this is checked by looking up the combined remotegroupcoursefield and remotegroupidnumberfield in the in-memory list of groups) then...

Make sure the group name doesn't clash with an existing group (without an idNumber)
If it does generate a warning: "Group "$groupName" was not imported because an existing group exists with the same name but no idNumber"
Continue

Create new group
Use value from remotegroupidnumberfield for idNumber
Use value from remotegroupnamefield (if set) for name - otherwise use value from remotegroupidnumberfield
Use value from remotegroupdescriptionfield (if set) for description - otherwise leave description blank
Set the courseId to the current courseId

If there is a value in the remotegroupgroupingfield then...
Look in the database for grouping where the remotegroupgroupingfield is equal to the value in this remotegroupgroupingfield
If no such grouping is found
Generate a warning: "Group "$groupName" was not placed in grouping "$groupingfield" because no such grouping exists"
Otherwise (if we can find the grouping)...
Put the newly created group in the the grouping.

# Edit existing groupings that have changed
If external group does currently exist in internal db then
If group name from external table doesn't match current group name then
Update name
If groupdescriptionfield is set and group description from external table doesn't match current group description then
Update description
Remove this grouping from the in memory list of internal groupings

# Delete any previously imported groups which are no longer in external db
Iterate across all the groupings left in the in-memory list of existing groups
# use the courseid and groupidnumber to identify the group to delete
Delete the group membership records (i.e. details of which user were in this group)
Delete the group
# Don't do anything to the groupings - even if this was the last group in the grouping
# if the user wants have a grouping with no groups in then that's fine

}  

Sync Group Membership
# this will be in its own method call which will be called before enrolment sync

if ( remotegroupmembershiptable and remotegroupmembershipuserfield and remotegroupmembershipgroupfield ) {
  if localgroupfield is not "id" then check that remotegroupmembershipcoursefield is also defined
    If not generate an error: "When using $localgroupfield to identify groups the remote group membership table must include the course."
    Return

  Query the remote database for a list of group memberships
  For each remote group membership record...
    Identify the group
    If we couldn't identify the group
      generate an error
      continue

    Identify the user
    If we can't identify the user
      generate an error
      Continue

    See if this membership is already recorded in the internal db (for any component)
    If not then
      Create a group membership record
      Set the component to be "enrol_database" component

    Record this groupmembership in an in-memory list of "seen" items (keyed on a combination of internal User ID and internal Group ID)
Query the database for all group membership records in the internal database belonging to the "enrol_database" component

For each of these group membership records...
  Look up the membership details (combination of internal User ID and internal Group ID) in the list of "seen" items then
  If the record doesn't exist in the seen items list then...
    Delete the groupmembership record
    # Don't do anything to the group - even if this was the last member
    # If the user wanted the group deleted then they would have removed it from the remote group table