Cameron L Palmer
October 21, 2006

SQL Table Layout

Table Key
* Indicates a Primary Key
^ Indicates a Foreign Key

Department Table
// For the time being the department table only contains the department codes and the titles of the departments. In the future it may be desirable to attach the college associated with each department.

dept_id*	***title***
VARCHAR()	VARCHAR()
CSCE	Computer Science and Computer Engineering
MATH	Mathematics

Course Table
// The course_id is initially conceived as a combination of the department and course number field but that might be changed.

course_id*	***dept_id^***	***course_no***	***title***
VARCHAR()	VARCHAR()	INT()	VARCHAR()
CSCE1010	CSCE	1010	Introduction to Computer Science

Term Table
// The term_id is based upon UNT's EIS numbering scheme.

term_id*	***semester***	***year***
INT()	VARCHAR()	INT()
1071	Spring	2007

Class Table
// class_id should be a unique autogenerated key. Although it seems retarded it may be possible that an instructor could want to have two or more gradebooks for a class.

class_id*	***course_id^***	***section***	***term_id^***	***cutoffs***	***categories***	***hide***
INT()	VARCHAR()	INT()	INT()	VARCHAR()	VARCHAR()	TINYINT()
	CSCE1010	002	1071	A=90,B=80,C=70,D=60,F=50	Homework=10,Quizzes=10,Exam I=20,Exam II=20:Final Exam=20,Term Project=20	0

Low-level Functions

```
bool dept_create(dept_id, title)
{
  if (!dept_exists(dept_id))
    // Query string should contain properly formatted SQL
    query := 'INSERT dept_id, title INTO dept'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool dept_edit(dept_id, title)
{
  if (dept_exists(dept_id))
    // Query string should contain properly formatted SQL, will want to update
    // only changed information
    query := 'UPDATE dept SET title WHERE dept_id={dept_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool dept_delete(dept_id)
{
  if (dept_exists(dept_id))
    // Query string should contain properly formatted SQL
    query := 'DELETE FROM dept WHERE dept_id={dept_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
array dept_get(dept_id)
{
  if (dept_exists(dept_id))
    // Query string should contain properly formatted SQL
    query := 'SELECT * FROM dept WHERE dept_id={dept_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return result
  else
    return failure
}
bool dept_get_all()
{
  query := 'SELECT * FROM dept'
  results := mysql(query)
  return results
}
bool dept_exists(dept_id)
```

```
{
  result := dept_get(dept_id)
  if (result != 0)
    return success
  else
    return failure
}

******

bool course_create()
{
  if (!course_exists(course_id))
    // Query string should contain properly formatted SQL
    query := 'INSERT course_id, dept_id, course_no, title INTO course'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool course_edit(course_id)
{
  if (course_exists(course_id))
    // Query string should contain properly formatted SQL, will want to update
    // only changed information
    query := 'UPDATE course SET dept_id, course_no, title WHERE
course_id={course_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool course_delete(course_id)
{
  if (course_exists(course_id))
    // Query string should contain properly formatted SQL
    query := 'DELETE FROM course WHERE course_id={course_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
array course_get(course_id)
{
  if (course_exists(course_id))
    // Query string should contain properly formatted SQL
    query := 'SELECT * FROM course WHERE course_id={course_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return result
  else
```

```
    return failure
}
array course_get_all()
{
  query := 'SELECT * FROM course'
  results := mysql(query)
  return results
}
bool course_exists(course_id)
{
  result := course_get(course_id)
  if (result != 0)
    return success
  else
    return failure
}

******

bool term_create(term_id, semester, year)
{
  if (!term_exists(term_id))
    // Query string should contain properly formatted SQL
    query := 'INSERT term_id, semester, year INTO term'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool term_edit(term_id)
{
  if (term_exists(term_id))
    // Query string should contain properly formatted SQL, will want to update
    // only changed information
    query := 'UPDATE classes SET semester, year WHERE term_id={term_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool term_delete(term_id)
{
  if (term_exists(term_id))
    // Query string should contain properly formatted SQL
    query := 'DELETE FROM term WHERE term_id={term_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
array term_get(term_id)
{
```

```
  if (term_exists(term_id))
    // Query string should contain properly formatted SQL
    query := 'SELECT * FROM term WHERE term_id={term_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return result
  else
    return failure
}
array term_get_all()
{
  query := 'SELECT * FROM term'
  results := mysql(query)
  return results
}
bool term_exists(term_id)
{
  result := term_get(term_id)
  if (result != 0)
    return success
  else
    return failure
}

******

int class_create(title, dept, course, section, term, year, cutoffs[])
{
  if (!class_exists(dept, course, section, term, year))
    // Query string should contain properly formatted SQL
    query := 'INSERT title, dept, course, section, term, year, categories, cutoffs
INTO classes'
    result := mysql(query)
    if (result != success)
      return failure
    return class_id
  else
    return failure
}
bool class_edit(class_id, title, dept, course, section, term, year, cutoffs[])
{
  if (class_exists(class_id))
    // Query string should contain properly formatted SQL, will want to update
    // only changed information
    query := 'UPDATE classes SET title, dept, course, section, term, year,
categories, cutoffs WHERE class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool class_delete(class_id)
// We never want to delete things in most of these classes.
// If there were a large number of entries in the database it could wreak havoc to
delete a class. So instead we will set a flag to hide a class. If a teacher really
```

```
wants a class to disappear we might need a separate front end that is dedicated to
hazardous operations
{
  if (class_exists(class_id))
    // Query string should contain properly formatted SQL
    query := 'UPDATE classes SET hide=1 WHERE class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
array class_get(class_id)
{
  if (class_exists(class_id))
    // Query string should contain properly formatted SQL
    query := 'SELECT * FROM classes WHERE class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return result
  else
    return failure
}
array class_get_all()
{
  // Query string should contain properly formatted SQL
  query := 'SELECT * FROM classes'
  results := mysql(query)
  if (results != success)
    return failure
  else
    return results
}
bool class_exists(class_id)
{
  result := class_get(course_id)
  if (result != 0)
    return success
  else
    return failure
}

******
// In this section I use an PHPism, explode and implode, for string to array operations
// explode() means split a string by a string and return an array
// implode() means split an array by a string and return a string

bool class_category_create(class_id, category, value, rank)
{
  if (!class_category_exists(class_id, category))
    // Query string should contain properly formatted SQL
    results := class_category_get_all(class_id)
    new_array[] := array_merge(array_slice($results, 0,
rank),array({category}.'='.{value}),array_slice($results, rank))
    categories := implode(“,”, new_array)
```

```
    query := 'UPDATE classes SET categories={categories} WHERE
class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool class_category_edit(class_id, category, value) // Editing value
{
  if (class_category_exists(class_id, category))
    // Query string should contain properly formatted SQL
    results := class_category_get_all(class_id)
    i = 0
    while (category != grep('/{category}=\d/', results[i]))
      i++
    results[i] := category.”=”.value
    categories := implode(“,”, results)
    query := 'UPDATE classes SET categories={categories} WHERE
class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool class_category_name_edit(class_id, category, new_name) // Editing name
{
  if (class_category_exists(class_id, category))
    // Query string should contain properly formatted SQL
    results := class_category_get_all(class_id)
    i = 0
    while (category != grep('/{category}=\d/', results[i]))
      i++
    results[i] := new_name.”=”.value
    categories := implode(“,”, results)
    query := 'UPDATE classes SET categories={categories} WHERE
class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool class_category_rank_edit(class_id, category, rank) // Editing position
{
  if (class_category_exists(class_id, category))
    // Query string should contain properly formatted SQL
    results := class_category_get_all(class_id)
    i = 0
    while (category != grep('/{category}=\d/', results[i]))
      i++
    temp := results[i]
    // array_shift pops elements off the front of the array
    j = 0
```

```
    while (results)
      if (j = rank)
         new_array[] := temp
      else if (j == i)
        array_shift(results)
      else
        new_array[] := array_shift(results)
      j++
    categories := implode(“,”, results)
    query := 'UPDATE classes SET categories={categories} WHERE
class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
bool class_category_delete(class_id, category)
{
  if (class_category_exists(class_id, category))
    // Query string should contain properly formatted SQL
    results := class_category_get_all(class_id)
    // The magic function delete_array_element_matching returns an array
    categories := implode(“,”, delete_array_element_matching(category))
    query := 'UPDATE classes SET category={categories} WHERE class_id={class_id}'
    result := mysql(query)
    if (result != success)
      return failure
    return success
  else
    return failure
}
string class_category_get(class_id, category)
{
  if (class_exists(class_id))
    // Query string should contain properly formatted SQL
    query := 'SELECT categories FROM classes WHERE class_id={class_id}'
    result := mysql(query) // returns a string
    result_string := grep('/,*({category}=\d+),*/', result)
    if (result != success)
      return failure
    return result
  else
    return failure
}
array class_category_get_all(class_id)
{
  // Query string should contain properly formatted SQL
  query := 'SELECT categories FROM classes WHERE class_id={class_id}'
  results := mysql(query)
  results_array := explode(“,”, results)
  if (results != success)
    return failure
  else
    return results_array
}
bool class_category_exists(class_id, category)
```

```
{
  result := class_get(course_id, category)
  if (result != 0)
    return success
  else
    return failure
}

******

array mysql(query)
{
  connect_database()
  results := perform_sql_query(query)
  disconnect_query()
  return results
}
```

High Level Functions

```
// Note – In SRD we called these courses, but now they are called classes
void html_class_mgmt() // SRD Page 5 Figure 2
  Render HTML using data from low-level functions
  Clicking on 'Add a new course' will call html_course_create()
  Edit or delete section will call class_get_all()
  A delete link will exist next to each course that will call class_delete() which
doesn't actually delete anything but it will make it go away.

void html_class_category_mgmt() // SRD Page 6 Figure 3
  Render HTML using data from low-level functions
  After making changes we submit call to class_category_edit()

void html_class_create() // SRD Page 4 Figure 1
  Render HTML using data from low-level functions
  After gathering information on submit call class_create()

void html_class_edit() // SRD Page 4 Figure 1
  Render HTML using data from low-level functions
  After gathering changed information on submit call class_edit()
```