Open Administration for Schools 4.75

Developer Documentation 4.75

Les Richardson

July, 2010

Contents

Introduction

Open Administration for Schools will be abbreviated as **Open Admin** or **OA** throughout this document. OA is a loosely coupled system that is designed to be simple to update and develop for. It is designed to be run with 3 virtual web sites per school and a couple of additional websites that may be used for a division wide installation or with an individual school. However, this is entirely up to the school and only parts may be used as desired.

OA consists two main types of files:

- HTML files – these files are simply a series of links organized into groups for demographics, attendance, discipline, etc. and point to the matching script and may pass values to it.
- Perl (CGI) scripts – that are called from the HTML files and create links back to those web pages. The perl scripts use the perl DBI interface module to talk to a database (MySQL or PostgreSQL) to store or retrieve data. They may also run external programs such as pdflatex to generate PDF reports or external XML parsers to parse XML encoded data files.

Directory Overview

The distribution, as downloaded, consists of the following directories:

1. **docs** - empty; documentation is now available by separate download to keep sizes of the downloads small.
2. **global** - contains only a single configuration file, global.conf, to configure multiple schools together.
3. **iep** - contains the special education application which runs as a separate website.
4. **school** - contains the main websites for a single school.
5. **sis** - a single site for division wide use. Poorly used. Not much interest.
6. **utility** - contains files to help with installation.

The main directories of interest are **school** and **iep** which consist of directories which contain HTML files or perl scripts (CGI scripts).

The **school** directory contains files for the *main administration site* (used by secretaries and administrators), the *teacher site* for teachers, and a *parent site* for parents and students.

The directories are:

- **admin** - HTML files for the main administration website.
- **cgi** - Perl (CGI) scripts for the main administration website.

- **tadmin** - HTML files for the teacher website.
- **tcgi** - Perl (CGI) scripts for the teacher website.
- **padmin** - HTML files for the parent site.
- **pcgi** - Perl (CGI) scripts for the parent site.
- **etc** - stores configuration files that are used by the perl scripts.
- **lib** - contains libraries used by the perl scripts.
- **templates** - contains template files used by the perl scripts for input forms and reports. This allows the student and staff tables to be extended with additional fields. These fields are then added to the templates.

Directory Details - Main Administration Site

The main administration site uses the files in the **admin** directory and the **cgi** directory. The files in *admin* are:

- **index.shtml** - main index page.
- **attendance.html** - attendance page.
- **discipline.html** - discipline page.
- **fees.html** - fees system, also lunch program, locker system.
- **schedule.html** - scheduling, student/staff location search, transportation.
- **export.html** - export/import data and Saskatchewan SDS page.
- **repcard.html** - report card system, subjects, transcripts and subject enrollment.
- **eoy.html** - start/end of year page; manage staff, school dates, preregistration.
- **help.html** - help system, links to documentation.
- **admin.css** - the single CSS stylesheet that controls look and feel of the website.
- **entrynotes.html** - help file for student entry. Linked from index.shtml.
- **favicon.ico** - an empty file (0 bytes) present so that it's absence doesn't clog the error log files.

It also contains the following subdirectories:

- **download** - a directory in webspace into which pdf reports, etc. are copied by scripts so that users may download them. It should be cleaned out on a regular basis by cron scripts.

- **images** - the location of the school logos (full size for main page(logo.gif) and a thumbnail(logotn.gif) for the subpages).
- **js** - javascript libraries.
- **pic-big** - full size student pictures (mugshots). Size set on import by settings in etc/image.conf.
- **pic-sm** - thumbnail size student pictures.

The main admin site cgi script directory (**(cgi)** contains all scripts for the main site.

It also contains the following directories:

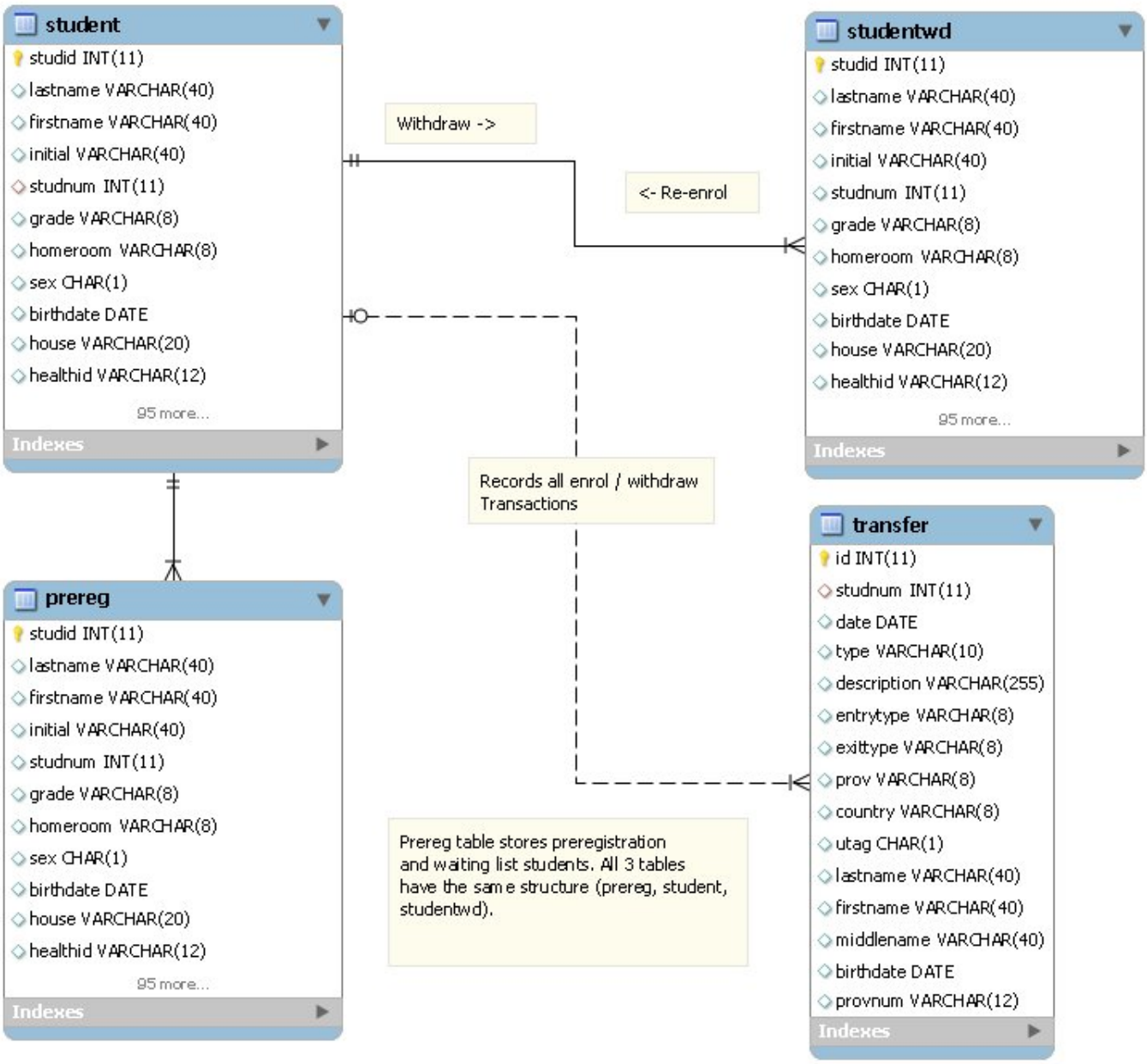
- **attendance** - All attendance management and reporting scripts.
- **discipline** - discpline scripts called from discipline page.
- **dbfimport** - import scripts from SIRS 3 software (SIS)
- **entry** - student enrollment and withdrawal scripts.
- **eoy** - start/end of year scripts such as staff and date managment.
- **export** - export scripts for exporting data in a variety of formats.
- **fees** - fees subsystem - accounts receivable.
- **image** - image management scripts for student pictures (mugshots).
- **inac** - Canadian schools INAC (Indian Northern Affairs Canada) reporting system to federal government for First Nations schools.
- **ldap** - LDAP Management scripts - manage users on external LDAP server
- **locker** - manage lockers and locks.
- **meta** - the meta scripts to edit and update the meta table. This table controls the type of entries for fields in the student and staff forms and reports.
- **old** - all directories may have older unused scripts stored in these named directories.
- **prereg** - preregistration and waiting list scripts.
- **repcard** - report card system including printing of report cards, subject management, and subject enrollments, etc.
- **repcard_pk** - Pre Kindergarten report card scripts. Actual reporting itself is in the report card script in repcard folder.
- **sasklrn** - scripts to update Sask Learning with the latest local data and to Query the provincial database as well.
- **schedule** - student scheduling scripts including scripts to locate students and staff at particular times, etc.

- **staff** - staff management scripts for the staff and staff_multi tables.
- **usermanage** - manage user accounts on external servers.
- **xlat** - translation scripting to read other scripts and html files and generate phrases to be translated. Also rewrites scripts with new

The **SaskEd management system** has scripts in **cgi/sasklrn**, but is not freely distributed since it it geographically limited in area to only Saskatchewan, and this helps to pay for development.

Data Tables / Scripts Overview

Demographics



The main demographics information is stored in several student tables:

The **student** and **studentwd** tables are the tables that hold student information and are identical in structure. The studentwd table holds withdrawn students. When the student withdraws, their record is moved from the student to the studentwd table. In the event of re-enrollment, the record is just moved back into the student table. The **studentall** table is a virtual table that is the joining of the two student tables. It is effectively a *view* but in MySQL 4.x, this is called a merge table.

The *studnum* field uniquely identifies the student. Within a school division blocks of student number are assigned to each school to keep the student number (studnum) unique within the division. For individual stand alone schools, this is not an issue. It starts a 100 by default.

The **transfer** table stores enrollment change information. Every time a student enrolls or withdraws from the school, one record is added to the table.

The *staff* table) is used to identify the teachers, TA/EA's (teacher aides or educational assistants), and other teaching staff members. Each person is uniquely identified by their *userid* field. Each person should have this field entered. The *homeroom* field will match the homeroom field in the student records. The *doatt* (Do Attendance) field indicates which teachers do attendance (and are tracked for entry of daily attendance). The *certif* field holds the teaching certification number for use with the Saskatchewan SDS system. (or other provincial or state based numbering system).

The main administration site has the following scripts in **cgi**:

- **emailgroup.pl, emailgroupSM.pl** - email messages to groups of parents and/or students. The SM version uses the Sendmail interface if you have an internal email server.
- **labels.pl** - label printing script with a variety of formats and layouts. Also runs from teacher site.
- **rptbillet.pl** - report on locations where children stay in the result of bad weather and other situations where they cannot be transported home.
- **rptbirthday.pl** - birthday report with a variety of options. Replaces the previous 2 separte reports.
- **rptcustomclasslist.pl** - A highly customizable report to print student lists for various types of recording. Choose between grade or homeroom, column widths, etc.
- **rptcustomstafflist.pl** - A similar report with many output options for printing staff lists.
- **rptdemoconf.pl** - Demographic confirmation report. Based on a template. It is normally printed to be sent home and signed by parents to confirm student information.
- **rptenrol.pl** - student enrollment report for any particular date. (by reading current enrollment and transfer table changes)
- **rptenrolmon.pl** - monthly enrollment change report with breakdowns by ethnic categories.
- **rptethnic.pl** - report ethnic numbers by grade.
- **rptfamily.pl** - family grouping report based on phone number. Will be replaced by a better solution when demographics system is redesigned.
- **rpthouseleague.pl** - house league (intramural teams) listing of student membership by homeroom.

- **rptmedical.pl** - a medical report that will do text searches in the medical field. New upcoming versions will have better reporting along with a new medical subsystem.
- **rptphone.pl** - a simple web based form to list student phoning information.
- **rptreligion.pl** - student report showing student religion by grade.
- **rptrelsacra.pl** - a religious sacraments report showing religion fields for students.
- **rptreserve.pl** - a report listing students and their home reserve if field is filled in.
- **rptstudrost.pl** - a pdf report listing complete student demographic information (one or two records per page). It is based on a template and is extensible.
- **rptsumage.pl** - a report that gives age and gender breakdowns in the classes and grades.
- **rptsumenrol.pl** - summary school enrollments by grade.
- **rptsumethnic.pl** - report on summary ethnic origins of students by grade.
- **rptsumrel.pl** - a summary religion report by grade.
- **rpttrack.pl** - a track report breaking students down by age group based on an entered aging date (ie. August 30, November 1, etc.)
- **rpttransmon.pl** - a monthly enrollment change report (transfers are enrollment changes and are stored in the transfer table).
- **rptyoungest.pl** - a report listing those students who are indicated to be the youngest in their family. This is useful since letters home to parents are frequently sent with the youngest.
- **staffpopmulti.pl** - temporary script to convert the staff table to the new multi-table format (version 4.75) . To be Removed in later versions.
- **studed.pl** - a script that allows editing of student demographic information. It uses a template from the templates directory (and is customizable as a result).
- **studeled.pl** - a report of all students in all student tables - current, withdrawn or pre-registered/waiting list. It then links to studed.pl to allow editing of student information. It doesn't allow deletion of students since this requires a withdrawal.
- **studsearch.pl** - a student search function to search demographic records by number, name or initials. I like initials the best.
- **studview.pl** - a templated full demographic record view of student demographic information.
- **templatecreate.pl** - a template creator for the student roster report (rptstudrost.pl)
- **viewlang.pl** - language profiency reports

When students withdraw, they are moved from one table (student) to another (studentwd) When they re-enrol, they are moved back. A transfer table keeps track of all of these movements and all new enrollments, etc.

The entry system (enrol/withdraw) has the following scripts in **cgi/entry**:

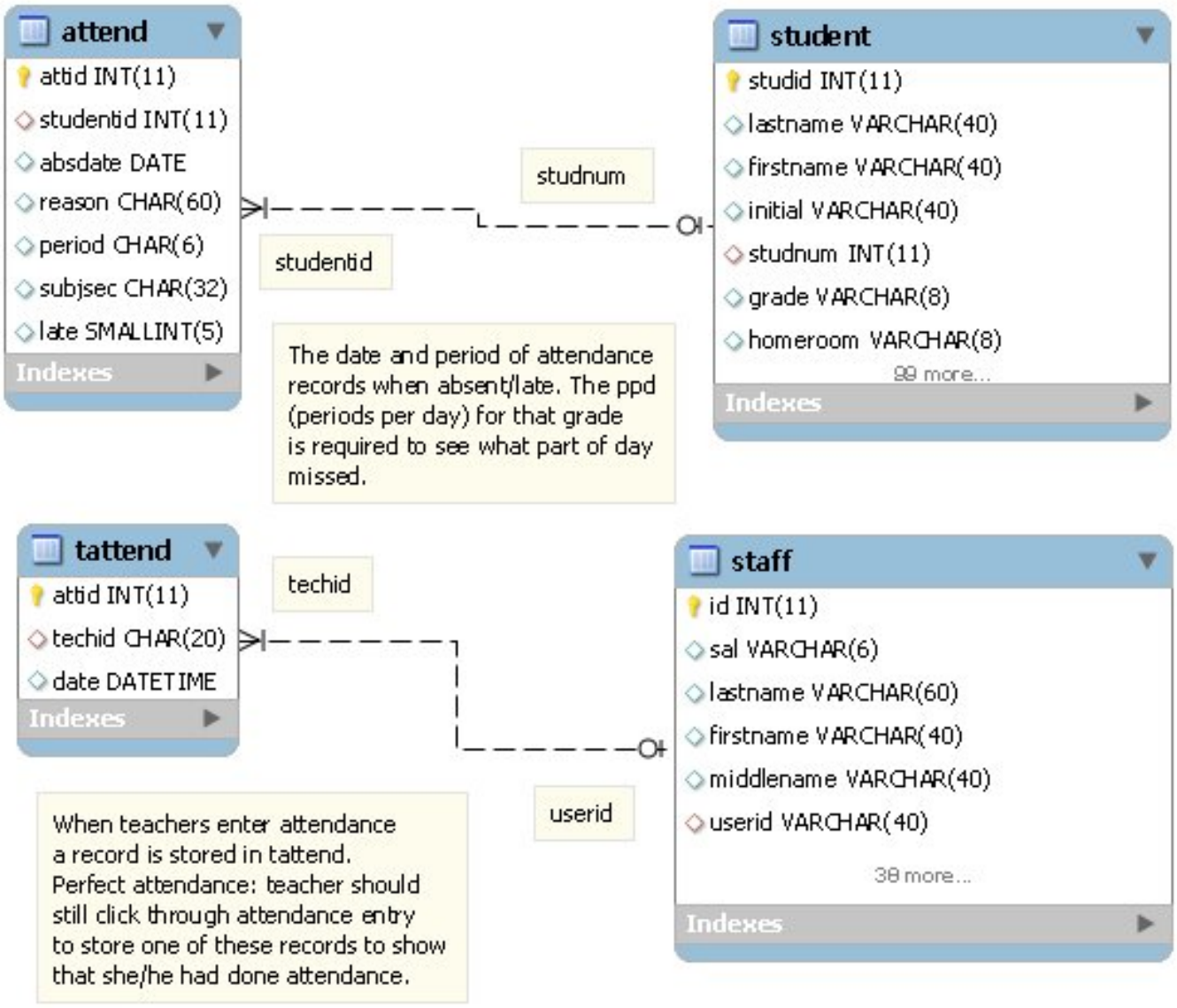
- **globenrol.pl** - called by student enrollment script (sentry0.pl). Copy demographics data, etc. from other school into local records.
- **globview.pl** - show withdrawn students in entire division.
- **reenrol.pl** - called by student enrollment script (sentry0.pl). Move student from withdrawn table (studentwd) into active student table (student).
- **sentry0.pl, sentry1.pl, sentry2.pl** - Student Enrollment Script; sentry0.pl - describes direction of student record flow (withdraw, enrol, re-enrol from withdrawn or global).
- **transadd0.pl, transadd1.pl** - Add a new transfer record; a blind script not called from anywhere. It would need to be entered from location bar in browser to run.
- **transchk.pl** - check transfer records for internal consistency, etc. Many checks done.
- **transdel.pl** - Delete transfer records; called by transview.pl
- **transed.pl** - Edit transfer records; called by transview.pl
- **transview.pl** - View transfer records; called from main page.
- **waitdel.pl** - delete waiting list entry; called by waitview.pl
- **waited.pl** - edit waiting list entry; called by waitview.pl
- **waitorder.pl** - alter ordering of the waiting list; called by waitview.pl
- **waitview.pl** - view waiting list
- **waitxfer.pl** - place students onto the waiting list.
- **withdraw.pl** - withdraw a student from school. Move his/her records to the withdrawn student table (studentwd). Called by the starting student enrollment script (sentry0.pl).

The start/end of year area has the date management scripts but also student reset scripts and permanent student deletion, password changes, configuration viewer, etc. It has the following scripts in **cgi/eoy**:

- **confview.pl** - view all configuration files for the school.
- **permdel.pl** - permanent deletion of student data (except values in transcript (course results) and transfer (enrol/withdraw) tables, which are long term storage). This includes their demographics record in studentwd (withdrawn student) table and also evaluation, discpline, and attendance.

- **pwdfix.pl** - fix any passwords with trailing cr/lf values. Not called anywhere; blind url. 2007 script.
- **pwdupd.pl** - update blank passwords. Out of date. Called from start/end of year page. Uses old apg method...
- **resetgrade.pl** - reset all student grade fields.
- **resethroom.pl** - reset all student homeroom fields.
- **resetselect.pl** - reset any selected student field. Very useful.
- **withdrawprom.pl** - Withdrawn promoted students.

The Attendance System



The Attendance system is stored in a couple of tables:

1. The **attend** table stores student attendance. There is one record for every period that a student **misses**. The student is identified by the *studentid* (aka *studnum* fields in other tables) field. This information is used to link together large numbers of tables. The *absdate* (absent date) and *period* field identify when the miss occurred. The *reason* and *late* fields indicate the reasons for absence to help categorize the records. The *subjsec* field identifies the subject / section (*subjsec*) missed for subject based attendance. This field is normally not used in elementary schools. The *late* field is not used by any functions, yet. The *id* field (called attid in this table; a design mistake) is used to uniquely identify each record to allow them to be edited or deleted in the event of an entry mistake. This id field is used in most other tables also, for the same reasons.

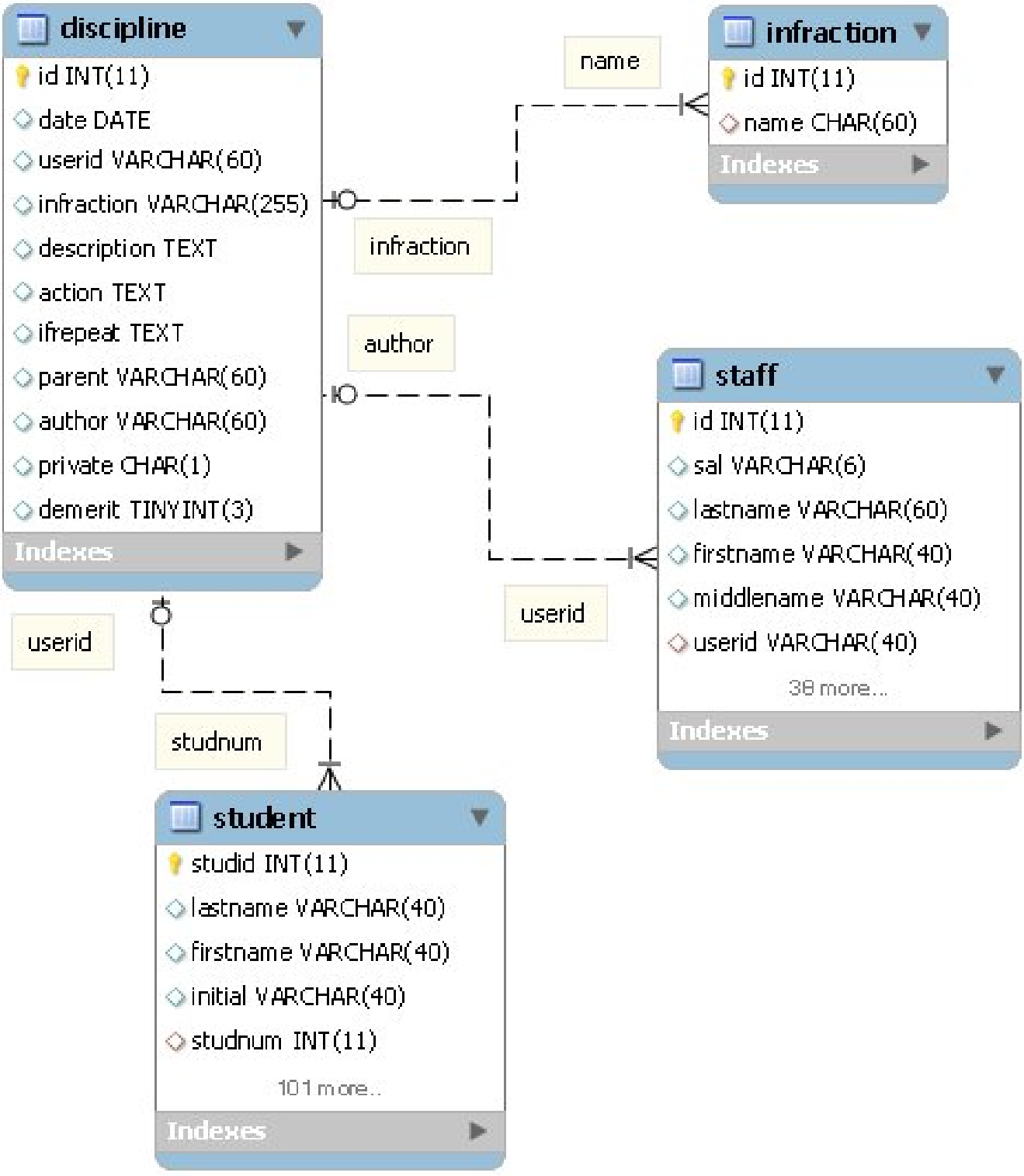
2. The **tattend** table is used to track teacher attendance entry. Every time a teacher enters attendance on the teacher site, a record is stored in this table. This will allow central office to ensure that every teacher has entered his/her attendance (even if perfect attendance) so that absences can be tallied, etc. This is necessary since OA only tracks "absences", not "presences". In many schools, this is not required.

The attendance system has the following scripts in **cgi/attendance**:

- **attaddall.pl** - Main attendance entry method for secretaries. Will do all grades, by homeroom or subject.
- **attcheck.pl** - checks attendance records for duplicate records, blank reasons and dates. It allows for fixing the errors also.
- **attdel.pl** - delete attendance record. Linked via the attdeled.pl which is a main menu script.
- **attdeled.pl** - display a list of attendance records, and then allow them to be edited or deleted. (via the attdel or atted scripts).
- **atted.pl** - edit attendance record. Linked via the attdeled.pl script.
- **attentrymd0.pl** - multiple day (md) attendance entry. Select weekly/biweekly, student groups, and periods.
- **attentrymd1.pl** - multiple day (md) attendance entry. Reason selection.
- **attentrymd2.pl** - multiple day (md) attendance entry. This stores the information in the attendance table (called attend).
- **attethview.pl** - view attendance by ethnic category. A summary report.
- **attpwrdel.pl** - attendance power delete. Allows rapid removal of attendance entered by mistake.
- **attrsnreset.pl** - reset the reason for an attendance record. The displayed reasons are for a particular date.

- **attscan.pl** - a large script to scan for students with attendance problems that will trigger discipline events. The settings in etc/admin.conf control the triggering of particular discipline events based on accumulated attendance points. A particular attendance event is assigned a particular point value. It will automatically add those discipline events and also generate PDF form letters for mailing home. The form letters template is in the *forms* directory.
- **attscanview.pl** - related to the attscan script above. It is passed a subject-section or a student number and will display the attendance record for those students along with their current attendance *points* values. A certain number of points will generate a discpline event.
- **attview.pl** - View attendance records. View by student or class.
- **chkattend.pl** - A script to check for teacher attendance entry (ie. that they've done their attendance).
- **chkstuatt.pl** - Check student attendance for those exceeding a certain number of days (input by user). Weakness is that it relies on embedded 'Absent Unexcused'.
- **rptattdaily.pl** - Daily attendance report with full stats at end, etc.
- **rptattmonth.pl** - Monthly attendance report.
- **rptattperf.pl** - Perfect attendance report. Find all students that have perfect attendance. Settings in the admin.conf file can allow certain attendance records to be ignored (ie. lates) depending on school policy.
- **rptattphone.pl** - A report to display phoning information for students absent/late for the day. Normally used by attendance officers, etc.
- **rptattprof.pl** - Attendance Profiles Report. A report to do simple attendance calculations and reports on all attendnace records. This is normally used by schools that are not yet using report cards (which contain attendance information also).
- **rptattstuday.pl** - Daily attendance report
- **rptattstudsubj.pl** - A cumulative report for the current term for each student by subjects.
- **rptattsubj.pl** - Attendance report by subject.
- **rptattyear.pl** - Yearly attendance report
- **rptcohort.pl** - Analysis of cohorts in terms of students at start of year compared to the same group at the end of the year.
- **rptinac.pl** - INAC (Indian Northern Affairs Canada) attendance report. Basically a monthly attendance report.

The Discipline System



The discipline system uses two tables:

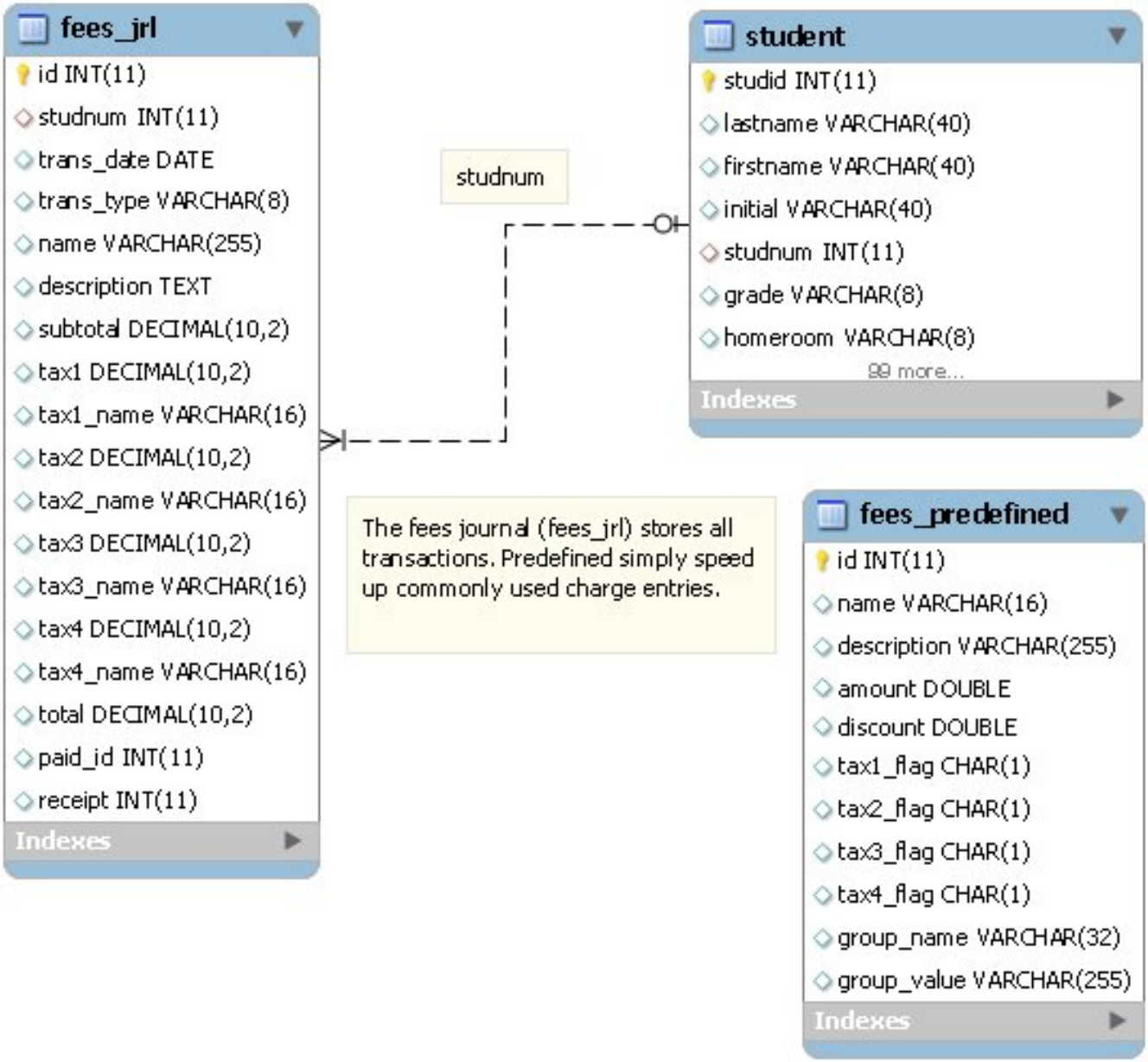
1. The **infraction** table which stores the descriptive text for the different kinds of discipline events in the school.

2. The **discipline** table stores the discipline infractions for the students. There is one record for each infraction. The *userid* field is the student number field (*studnum*). The *author* field is the userid for a staff member. The *private* field is used to mark records only visible from the admin site, not the teacher site. The *parent* field indicates the nature of parental notification about the event. The *action* field indicates any action take due to the event, and the *repeat* field is the nature of the consequence in the event of a repeat of this behavior.

The discipline system has the following scripts in **cgi/discipline**:

- **catadd0.pl,catadd1** - adds new infraction categories
- **discadd0.pl,discadd1.pl, discadd2.pl** - Scripts to add discipline events.
- **discbulkdel0.pl, discbulkdel1.pl** - delete records by date and/or student.
- **discdatedel.pl** - delete records by date.
- **discdel0.pl, discdel1.pl** - record delete
- **discdeled.pl** - master discipline edit/delete script; calls other scripts
- **disced0.pl, disced1.pl** - Edit Discipline records
- **discvw.pl** - View Discipline records
- **infdel0.pl** - Lower level infraction delete; called by infdeled.pl
- **infdeled.pl** - Delete Infraction Categories
- **infvw.pl** - View current infraction categories.
- **reportsum.pl** - School Summary Discipline report
- **rptdiscstat.pl** - Summary report with statistics.
- **rptstud.pl** - Student Discipline report

The Fees System



This system is used to track student fees owing, and notify parents of amounts owing.

There are two tables used to implement this system:

1. The **fees_jrl** is the main fees journal table and holds all transactions.
    - **id** int(11) not null auto_increment - the normal id field to identify this record.
    - **studnum** int(11) default null - the student number field. This field identifies the student that this transaction applies to. This links to the student number (studnum) field in the student table.
    - **trans_date** date default null - the transaction date. Scripts allow the user to change

the date (if doing entries after the fact). They are not forced to make this todays date.

- **trans_type** varchar(8) default null - the transaction type. There are currently three possible values here:
  - **chg** - for a normal charge transaction where a student is charged a particular fee.
  - **pay** - a payment for a single charge or several charges.
  - **roa** - received on account for a payment where there is an overpayment or there is not sufficient payment to pay a change transaction completely.

  There is more below on the nature of these transactions.
- **name** varchar(255) default null - a short name describing this transaction such as *School Uniforms, Book Fees*.
- **description** text - a full unlimited length text description of the transaction (normally only possibly required for the charge transactions).
- **subtotal** decimal(10,2) default null - the value of the transaction before taxes. It is left null for payments and received on account (roa) transactions since there are no tax values.
- **tax1** decimal(10,2) default null - the value of a particular tax. These tax names and their rates are set in the fees.conf configuration value.
- **tax1_name** varchar(16) default null - the name of the tax as set in the fees.conf configuration file.
- **tax2** decimal(10,2) default null, **tax2_name** varchar(16) default null, **tax3** decimal(10,2) default null, **tax3_name** varchar(16) default null, **tax4** decimal(10,2) default null,

  tax4_name varchar(16) default null - same as the tax 1 values. OA has room for up to 4 taxes which seems to be the largest worst case scenario.
- **total** decimal(10,2) default null - the total value of the transaction including all taxes (the sum of the subtotal and tax1 to tax4 if present). For pay and roa transactions there are no taxes and only this field is used to store the value of the transaction. The subtotal field is left null in these transactions.
- **paid_id** int(11) default null - this field is filled in for charge (chg) transactions with the record id of the matching payment (pay) transaction. When this field is null for a charge transaction, this transaction is outstanding (ie. unpaid).

  Payment transactions hold the receipt number (obtained from the *receiptnumber* file in the etc configuration area) in this field.

  ROA (Received on Account) transactions are generated when the value of the payment is less than the value of the charge transactions. The transaction type (trans_type) of the record is *roa*. These records have the receipt id value in the description field and paid_id remains NULL until it is included with a payment and linked to a particular charge or charges.

2. The **fees_predefined** table is the table that stores predefined charges. It includes the following fields:

- **id** int(11) not null auto_increment - record id for this record.
- **name** varchar(16) default null - the short name of the transaction.
- **description** varchar(255) default null - a longer description for the transaction.
- **amount** double default null - the amount of the transaction.
- **discount** double default null - a discount rate (in percent) that could apply to this charge. This is currently unused in any script.
- **tax1_flag** char(1) default null, **tax2_flag** char(1) default null, **tax3_flag** char(1) default null, **tax4_flag** char(1) default null - flags to indicate that this tax applies to this charge. A non-null value indicates true (normally a one (1) is stored).
- **group_name** varchar(32) default null - the name of the group that these charges apply to. This may be *Grade or homeroom*.
- **group_value** varchar(255) default null - the value of the grouping. If for a grade this might be a 10 or if a homeroom is might be a value like *2RR*.

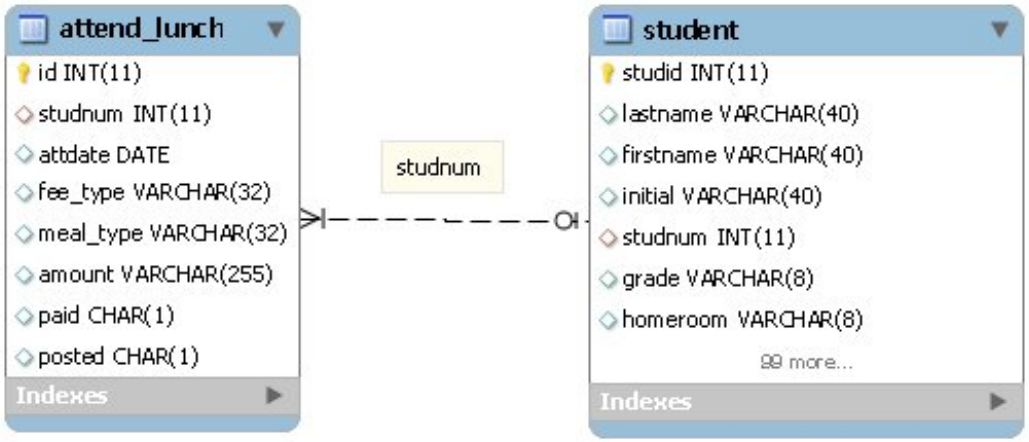
The last two fields indicate to which students this charge would apply (although this can be changed when applying the charge).

All of these scripts are located in the cgi/fees directory and are linked via the fees.html page on the main menu. They include:

- **Assess Fees (assessfees.pl)** - is used to charge fees to particular students or student groups. The fees can either be predefined fees or created from scratch. The fees are then applied to particular student groups.
- **View Assess Fees (assessview.pl)** - provides a simple view of all transactions in the fees_jrl table. The script just does a dump of all records.
- **Delete Fees (assessdeled.pl)** - delete assessed fees (ie. charges) that are not yet paid.
- **Change Paid Status (changepaid.pl)** - allows lunch fees to be marked as paid. An interim script only until summary lunch values can be posted into the fees system.
- **Export Monthly Summary (exportmonthjrl.pl)** - export summary transactions for a month into an external accounting system (csv format).
- **Print Outstanding Invoices (invoice.pl)** - used to print invoices for parents of amounts owing, payments made, etc. Only prints outstanding invoices. needs dimension updating for pdf generation.
- **Payments (payment.pl)** - add payments and print receipts.
- **Add Predefined Fees (predefadd.pl)** - add predefined fees.
- **View Predefined Fees (predefview.pl)** - view all predefined fees, the amounts, and who they apply to.
- **Print Receipts (receipt.pl)** - print receipts (also called directly from payments).

- **View Outstanding Fees (rptoutstanding.pl)** - from 'View Outstanding Fees' button; report any outstanding charges, all children, same family (HTML output)
- **transactview.pl** - show transactions for a certain student number.
- **rptpaid.pl** - same as rptoutstanding. Script is incomplete. Not linked to main page.
- **invoiceview.pl** - no longer used.(html output)

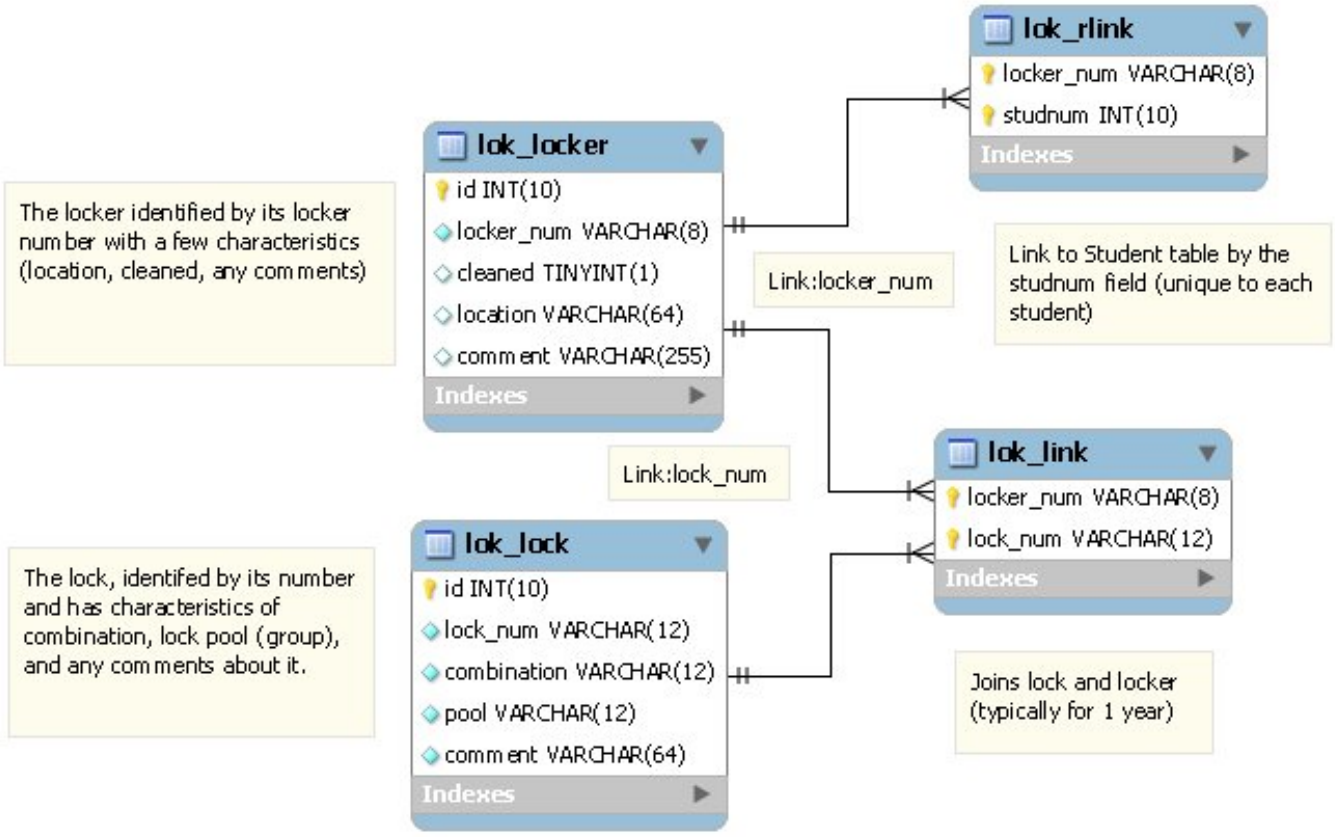
The Lunch System



The lunch system has the following scripts (also located in **cgi/fees**):

- **changepaid.pl** - change value of the paid field (ie. indicate paid).
- **lunchadd.pl** - add lunch entries for students or student groups (to indicate that they had a particular meal) for a particular date and also indicate whether paid, etc.
- **lunchrpt.pl** - a lunch attendance form report; prints a blank lunch entry form for use by lunch room staff.
- **lunchrptdue.pl** - a lunch report show fees owing; not used.
- **lunchsearch.pl** - name based lunch search with html output. Shows all lunch records (paid or unpaid).
- **lunchview.pl** - view lunch records, including ability to delete.

The Locker System



A system for managing locks and lockers.

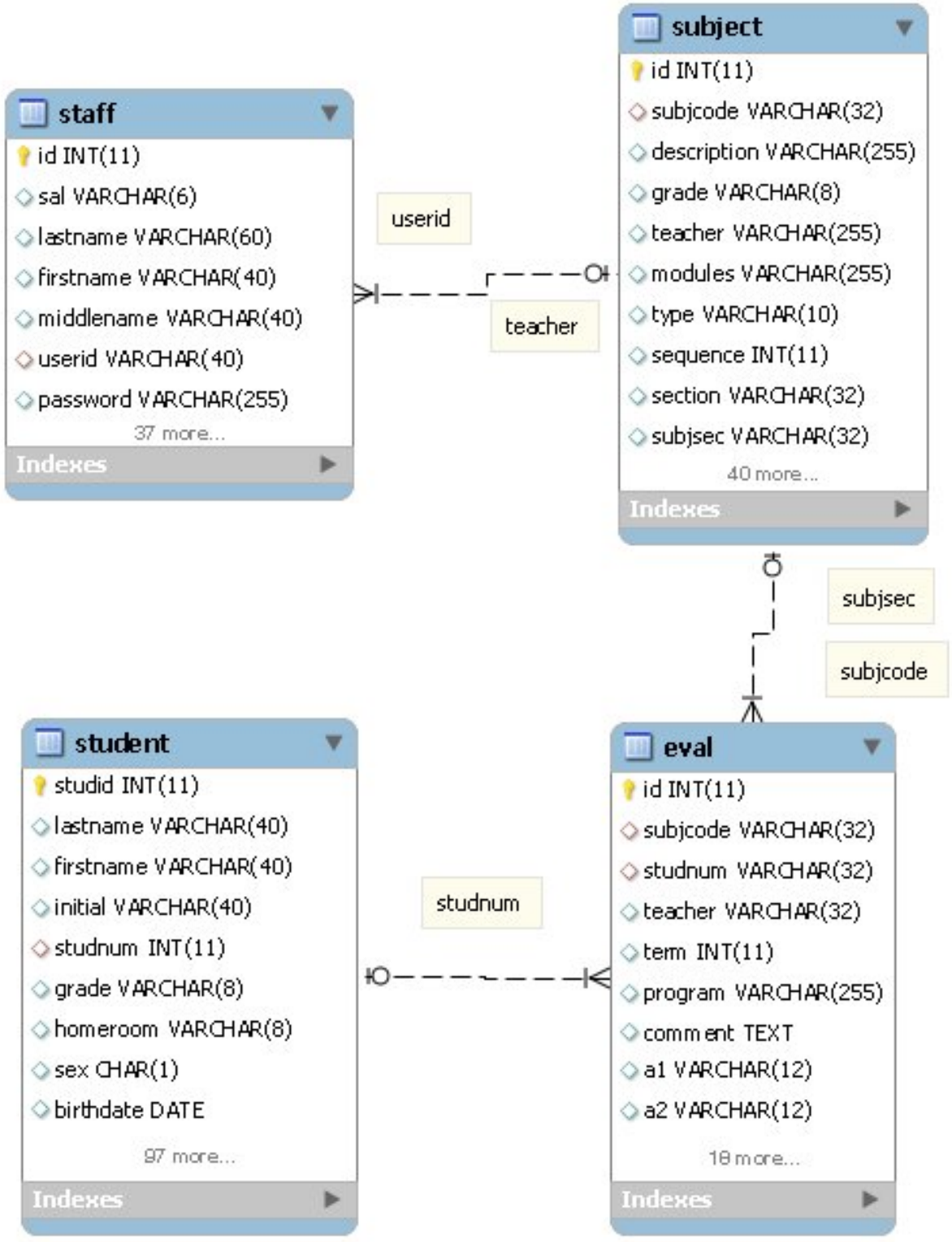
The 12 scripts (located in /cgi/locker) include:

1. lockeradd.pl - add lockers to the system.
2. lockerassign.pl - assign lockers to students.
3. lockerunassg.pl - unassign lockers.
4. lockeredit.pl - edit locker values.
5. lockersearch.pl - search for lockers based on criteria.
6. lockerview.pl - view locker values.
7. rptlocker.pl - full locker report.
8. lock_upload.pl - upload locks into the system (from csv file).
9. lockadd.pl - add locks into the system.
10. lockassign.pl - assign locks to lockers.
11. lockedit.pl - edit lock values.

12. lockview.pl - view lock values.

These are called from the locker area of the fees page.

The Report Card System



The Report Card System makes use of a couple of tables:

1. The **subject** table – this table stores information about a single course in each record. It lists the course subject, who teaches it, the course code and section number and the

starting and ending reporting periods. Each subject-section is identified by a unique subject-section code (called *subjsec*) which is the subject code, a hyphen, and a section code (ie. 8415-1)

2. The **eval** table – (or evaluation table) stores the teacher evaluations for a student in a particular class and section. When a student is enrolled, one record is entered for each reporting period in that class and section. For example, if there are 4 reporting periods (terms) for a particular subject/section, then 4 records are added for each student enrolled in that subject/section.

   This table stores not only the evaluation results but also the subject enrollment. When a student has records here, he/she is enrolled.

The subject table has the following fields:

1. The **teacher** is the userid and name of the teacher. This course will be listed under the teacher's name on the staff mark entry pages while entering evaluations. The name part will be removed in the future, since this is a design flaw. It will exactly match the userid from the staff table in the future.

2. The **paa modules** are course "sub-components" listed by Saskatchewan Learning. They only have meaning in Saskatchewan, and information from this field is not currently used by other parts of the admin system.

3. The **type** is an attempt to classify subjects on the basis of the kinds of entry values needed for evaluation. However, as of 2.00, you may leave this field blank.

4. The **sequence** is a number that controls the order with which this subject prints on a report card. If, for example, you wanted particular core subjects listed first, they would have lower numbers. It is suggested that one go up by tens (10,20,30) when entering sequence numbers to simplify the process of re-ordering subjects or adding in new ones.

5. The **web visible** field controls whether this subject could be visible on the web to authorized individuals (ie. Parents). This would, of course, mean only their own child and not all students taking the class. This is used by the administration software as of OA 1.50.

6. The **faculty** field lists which faculty offers this subject. It will help classify subjects for academic requirements. Currently unused.

7. The **location** field lists where this subject/section is offered. Currently unused, although this will change in the 2.x series.

8. The **mark scheme** is a text block that describes the evaluation process used by this subject. Currently unused.

9. The **20 objectives** fields describe different particular objectives, etc. that a teacher is using to evaluate his/her course. These will be printed on the report cards (normally on the administrative site) and are also present when entering student evaluations (on the teacher site). Normally the first objective field is used for numeric results, and later ones, if used, are more descriptive.

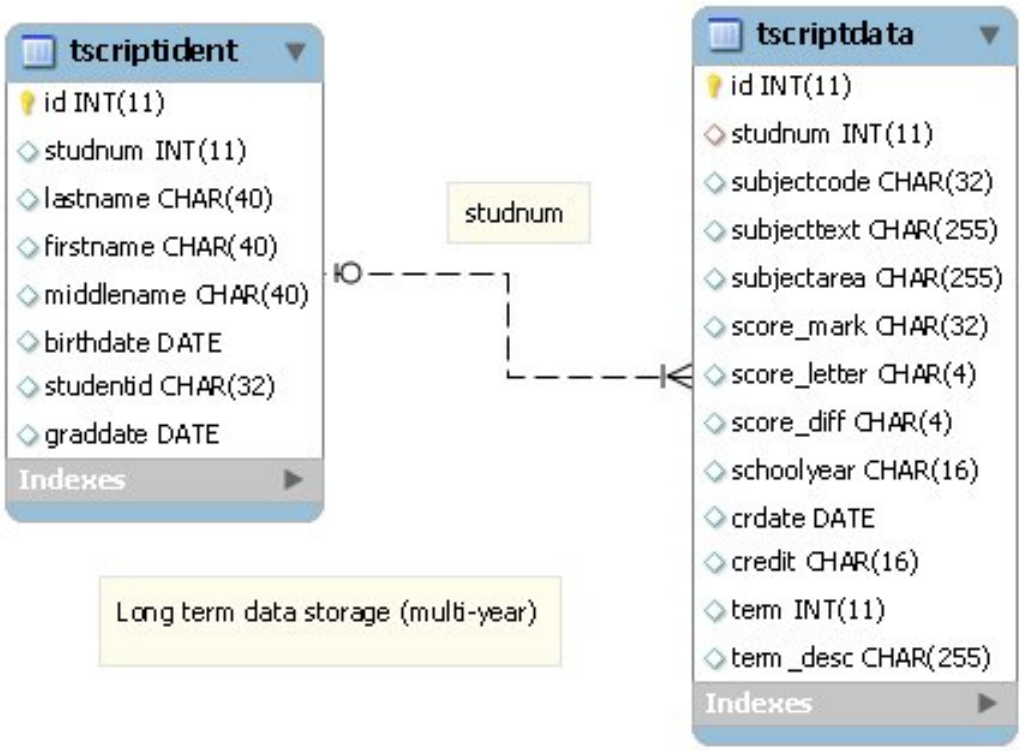
10. The **subject aliases** are small numbers that uniquely identify a subject/section and are used only for doing particular types of subject based attendance.

The report card system has the following scripts in **cgi/repcard**:

- **emptymarksview.pl** - View students with missing mark values. This is done before printing report cards so teachers may be notified about missing values.
- **enroladdall.pl** - add subject enrollments (ie. enrol students in subjects). This is the main script to do this.
- **enrolstudview.pl** - search for subject enrollments by student name, initials.
- **enrolview.pl** - view subject enrollment by student or subject.
- **evaldel.pl** - delete subject enrollments; called by evaldeled.pl
- **evaldeled.pl** - edit/delete subject enrollments.
- **evaldelstud.pl** - delete enrollments by student
- **evaled.pl** - edit subject enrollment values (single record); called by evaldeled.pl
- **evalmove.pl** - move subject enrollments from one homeroom to another; must be same grade and same subjects.
- **evalpwrdel.pl** - power delete subject enrollments; called by evaldeled.pl
- **evalpwrdel_rc.pl** - power delete alternate report card (PK report card)
- **rptemptysubjlist.pl** - pdf report of student without any subject enrollments. Older code; limited usage, imo.
- **rptenrolgrade.pl** - very simple older script; shows enrollment and each term record.
- **rptenrolsumm.pl** - from 'Enrollment Summary - Web' button. Shows subjects enrolled and the number of students. Also shows those subjects with no enrollment.
- **rptrank.pl** - 'rank students by gpa' button. Select by grade/homeroom and school year. html output.
- **rptrepcard.pl** - main report card and alternate report card printing script. Many options on the start page; it also pulls values from the repcard.conf file.
- **rptstudfinalmarks.pl** - 'Year End - Student Summary Marks' button. Needs updating to latest configuration values for pdf output. Start page has student selection, sort order, paper size, etc. Title is 'Cumulative Mark Report', and show school year, printing date, and then all the final marks.
- **rptsummark0.pl, rptsummark1.pl** - 'View Subject Marks' button; first script selects by grade, teacher, or subjects, and second generates the report (pdf format) with slanted objectives, etc.

- **rptsummarkend.pl** - 'Year End - Final Evaluation Report' button; select by grade or homeroom (and paper size). Prints all subjects and all students on a single page with only the first objective (ie. the mark value). Should be updated for pdf setting values.
- **rptsummarkstud.pl** - 'Student Term Marks Report' button. Title called 'Student Summary Marks'. Needs PDF update and title fix. Quite simple report.
- **rptsummxtab.pl** - 'View Crosstab of Subject Marks' button; title still called 'Summary Mark Report'. Has a start page for values. Can select by grade, homeroom or subject. Then papersize and term (blank=current). Has a report with slanted objectives and includes all objectives for all subjects for all students. This is normally landscape orientation.
- **saliasdel.pl** - Not used or linked from html pages. Student Alias delete.
- **saliasupd.pl** - Not used or linked from html pages. Student Alias update.
- **saliasview.pl** - Not used or linked from html pages. Student Alias view.
- **subjadd.pl** - add new subjects.
- **subjdel.pl** - delete subjects; called by subjdeled.pl.
- **subjdeled.pl** - edit/delete subjects.
- **subjed.pl** - edit subjects; called by subjdeled.pl.
- **subjview.pl** - view subjects; provides links to edit subject.
- **rptsubjteach.pl** - title: 'Teacher Subject Report'. HTML report showing teacher loads.
- **termed.pl** - Edit current term. This controls mark entry on the teacher site and posting from the gradebook, as well. A zero value disables. It also changes based on the multitrack variable, since there may be different values of term for different grade levels.

The Transcript System



The transcript system is designed to store student course information over multi-year time periods while the student is enrolled in school (and even after he/she has graduated or withdrawn/moved).

It consists of 2 tables:

1. **tscriptident** - this table stores the basic identity information about the student. There is only one of these records for each student.

   It includes:

   - **id** int(11) not null auto_increment - normal id field to identify the record.
   - **studnum** int(11) default null - the student number used to link this record to all of his/her course information records stored in the other table (tscriptdata). This field must be unique to this student since this is the most important number identifying the student in OA. This is the same field as found in all demographic, attendance, report card information, etc.
   - **lastname** char(40) default null - family name of student.
   - **firstname** char(40) default null - given name.
   - **middlename** char(40) default null - other name information.
   - **birthdate** date default null - birthdate, another key identifying piece of information.
   - **studentid** char(32) default null - another identifying number; characteristic of the educational jurisdiction. In Saskatchewan, this would be the provincial student

number. The field that is exported from the student demographics table (student) and placed in this field is a configured item found in the transcript.conf configuration file.

Other fields could be easily added to this table to store other information that is required about the student over the longer term (ie. perhaps other important number such as SSN (Social Security Number), etc.

2. **tscriptdata** - This table stores the actual course information. This information is read (along with the identity information) to create the student transcripts.

   - **id** int(11) not null auto_increment - record identifier.
   - **studnum** int(11) default null - links to the student identity table (tscriptident) in a one to many relationship. (One student - Many Courses, possibly).
   - **subjectcode** char(32) default null - the course code assigned to this subject by the school. This is called the subjsec (subject-section) code in the subject table and is the subjectcode in the eval table. They are one and the same value. They contain a subject code and a section code separated by a hyphen.
   - **subjecttext** char(255) default null - description of the subject (ie. Biology 12).
   - **subjectarea** char(255) default null - the area (science, math, etc.) of the course.
   - **score_mark** char(32) default null - the student's mark.
   - **score_letter** char(4) default null - the letter score for this mark. The mapping between a particular mark value and it's matching letter score is found in the transcript.conf file. They letter grades are not stored in the eval table (the table that stores the term results for each student in each course. The final term value is assumed to be the final overall mark.
   - **score_diff** char(4) default null - a difficulty value for this course. This is used to calculate the grade points for this course. The value is stored in the subject table for this course.
   - **schoolyear** char(16) default null - the school year in YYYY-YYYY format. (2007-2008)
   - **crdate** date default null - the date of this record. This is credit date; the date this credit was obtained. Currently, not much use if made of this since the school year is more helpful.
   - **credit** char(16) default null - the credit obtained in this course. Obtained from the subject table record for this course.
   - **term** int(11) default null - the term of this course. This represents the term of the eval table record that was posted. Not all that useful, except for record ordering of the transcript records (along with the year).
   - **term_desc** char(255) default null - the description of this term... values are mapped from the transcript.conf so that they describe the school's particular text for this period of time (since terms are smaller than Semesters (2 terms = 1 semester) or Quads (2 terms per quad; 4 quads per school year).

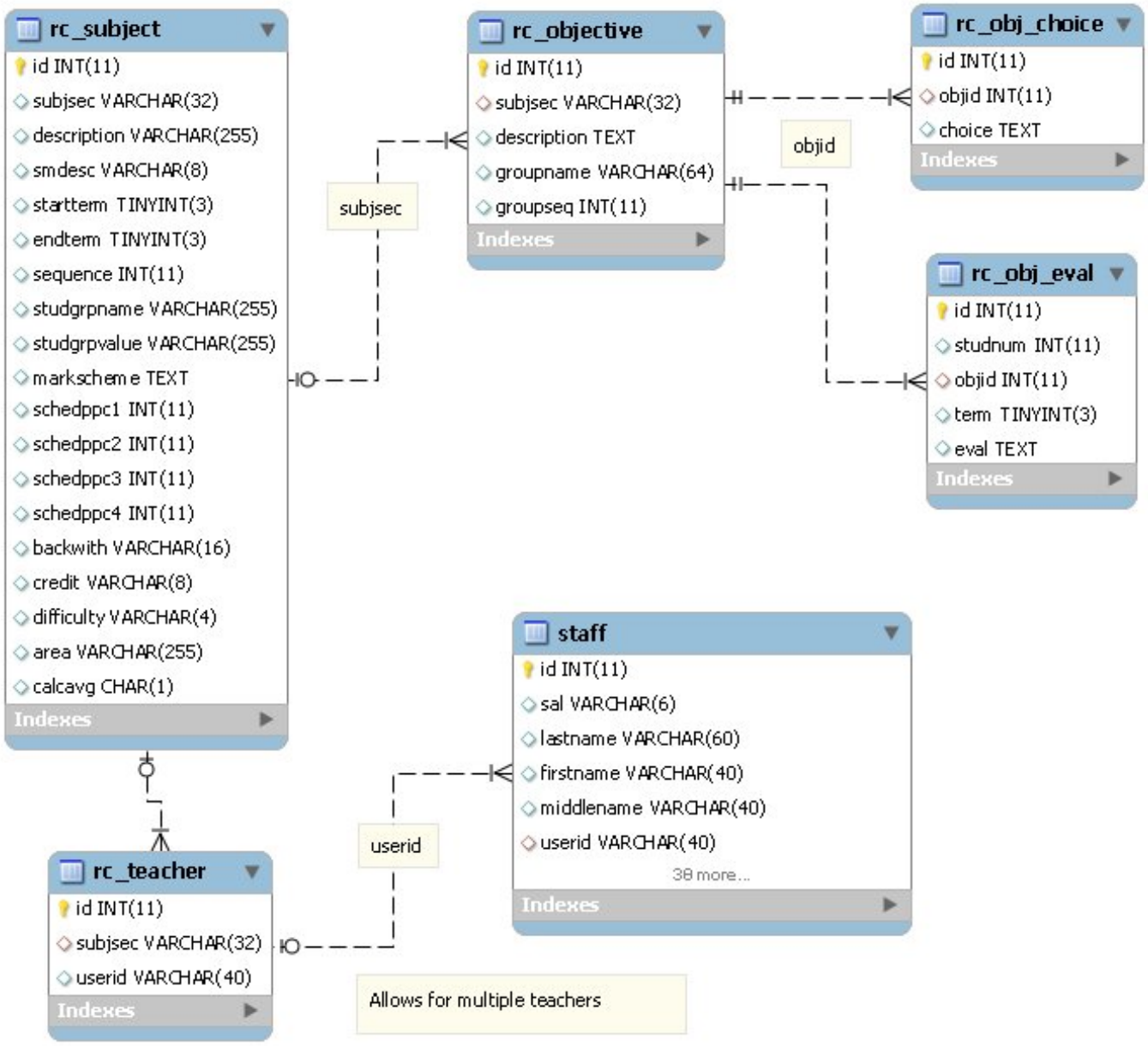
There are only a few scripts (3 currently) required to implement this system. As a result, they are all included in the cgi/repcard directory along with the other report card scripts. The scripts themselves should be read to understand their operation.

1. The **Transcript Post** script (**tscpost.pl**) - this script copies information from the current year's student evaluation (eval table) into the transcript tables (tscriptdata, tscriptident). It adds a transcript identity record if one doesn't already exist and then adds data records for the subjects selected. It will skip any existing records if they already exist. Existing records can be edited directly.
2. The **Transcript Edit/Delete** script (**tscdeled.pl**) - this script allows transcript records to be edited and deleted in the transcript data table. The student information table (tscriptident) is not affected since these records may remain even if there are no matching data records.
3. The **Transcript Report** script (**rpttranscript.pl**) - generates the actual PDF transcripts. It generates both an HTML output as well as the PDF output required.

The transcript system has the following scripts:

- **rpttranscript.pl** - print Transcripts (pdf/html format)
- **tscadd.pl** - Add transcript entries.
- **tscdeled.pl** - Edit/Delete transcript entries
- **tscpost.pl** - Post evaluation entries (ie. report card marks) into the transcript system.

The PK Report Card System

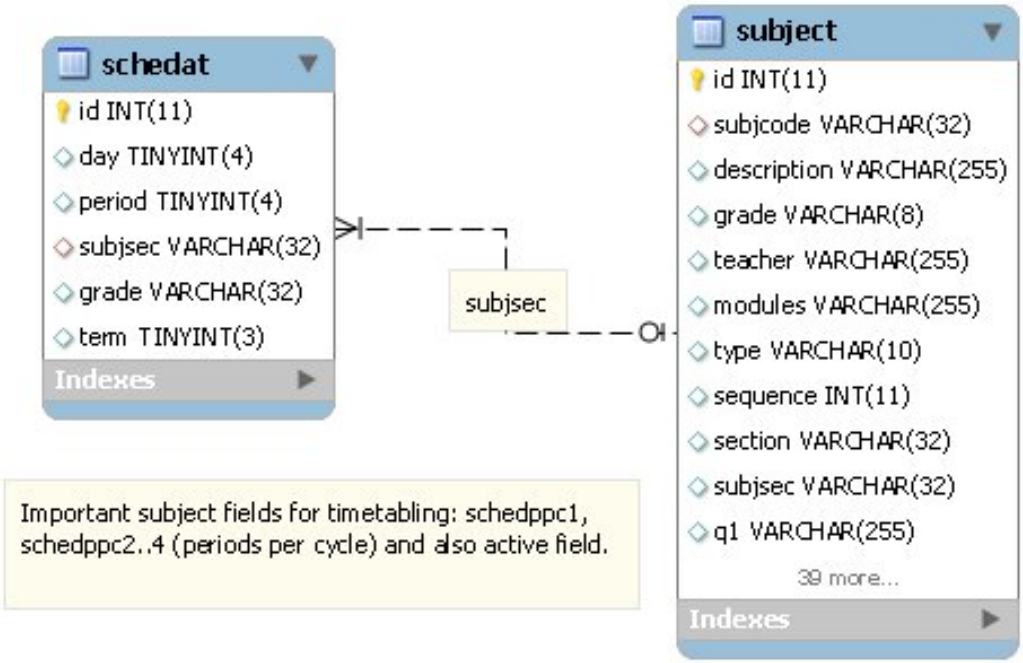


The pre Kindergarten report card has the following scripts in **cgi/repcard_pk**:

- **pkenroladd.pl** - enrol students in the PK subjects.
- **pkenrolview.pl** - view subject enrollment in PK subjects.
- **pkobjadd.pl** - add objectives to a PK subject.
- **pkobjview.pl** - view per PK subject objectives, delete objectives, and add or edit or delete choices for those objectives.
- **pksubjadd.pl** - add new PK Subjects
- **pksubjdeled.pl** - edit/delete PK Subjects

- **pksubjed.pl** - lower level PK subject edit; called by pksubjdeled.pl.

The Scheduling System



The schedule system has the following scripts in **cgi/schedule**:

- **exportfetdata.pl** - export xml data to the FET (Free Evolutionary Timetabler) timetable software.
- **importfetdata.pl** - import xml data from FET.
- **rptschedule.pl** -
- **schedadd.pl** - add timetable elements.
- **schedel.pl** - delete timetable elements.
- **schedit.pl** - edit timetables elements.
- **schedview.pl** - view student timetables (by grade)
- **tchschedview.pl** - view teacher timetable.

Subject scheduling scripts that edit data for FET timetabler use.

- **subjsched.pl** - Add/Edit Subject Periods per cycle and active or not. Used by timetabler to generate timetable based on constraints.
- **subjschview.pl** - View Subject Schedule entries. PPC and Active;

The Transportation System

Scripts for this system are located in /cgi/fees also.

- **transdaily.pl** - Daily transportation report, includes attendance.
- **transed.pl** - Edit transportation type (Bus, etc)
- **transedall.pl** - Edit all transportation values.
- **transrpt.pl** - Transportation report (pdf) - not yet templated.
- **transview.pl** - View Transportation fields.
- **wheresnelson.pl** - based on the children's book 'Where's Miss Nelson', this script gives teacher location based on their entered timetable.
- **whereswaldo.pl** - similarly, from the 'Where's Waldo' children book, this gives student location based on timetable.

The Export System

The export system has the following scripts in **cgi/export**:

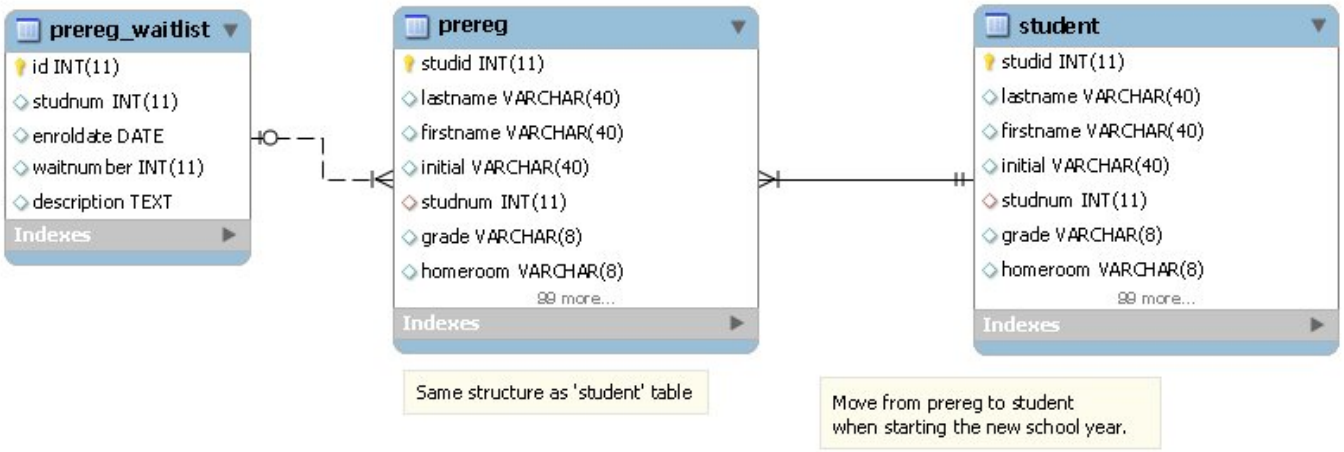
- **backupall.pl** - full database backup for download.
- **expaddress.pl** - export student and address information.
- **expcumfldr.pl** - export fields needed by the cumulative folder.
- **expcustom.pl** - export any selected fields.
- **expphoto.pl** - export information for school pictures.
- **exprepcard.pl** - export report card data.
- **expstaff.pl** - export staff data.
- **expstud.pl** - export full student data.
- **expstudent.pl** - export selected student data.
- **exptransfer.pl** - export transfer data (enrol/withdraw)
- **expuserpwd.pl** - export users and passwords.
- **expwdstud.pl** - export withdrawn students.
- **importCSV.pl** - import students.
- **importstaff.pl** - import staff.

Image Management

The image management system has the following scripts in **cgi/image**:

- **associate.pl** - this script will change the filename into a student number value (since the correct name is studentnumber.jpg (ie. 12345.jpg). It also updates the student pic field.
- **imageadd.pl** - upload image files (even in zip or tgz archives), extract, and resize into working directory for pictures. They should then be 'associated' to rename to correct student number value.
- **imageview.pl** - view and delete student images.
- **rptpicstudent1.pl** - new report for student pictures. pdf output suitable for color laser printing.
- **studpicview.pl** - another student picture viewer. Different options.
- **working** - working folder for image scripts

The Preregistration / Waiting List System



The **Preregistration System** is used when a school wants to:

1. Register kindergarten(K) and prekindergarten(PK) students before the end of the current school year.
2. Want to do homeroom and grade assignments before school end.

It uses 3 tables:

1. **prereg** - a clone of the student table. Preregistration entries (ie. K, PK) are stored here and then added to the student table later.

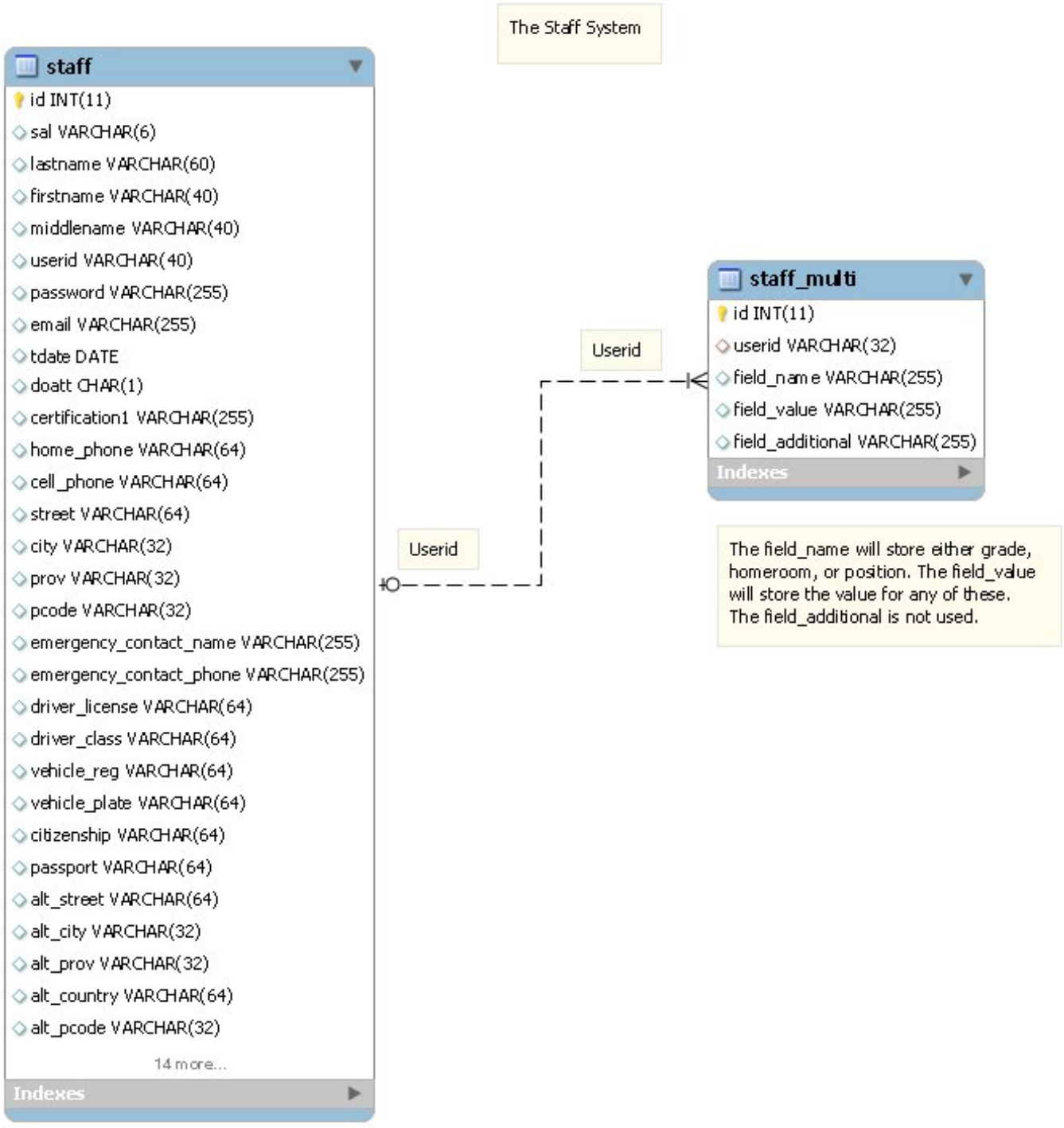
2. **preset** - updated values for homeroom and grade for students that will be used to update the student table later.

3. **prestaff** - clone of the teacher table. Stores new teaching homeroom assignments, etc. for pre-registration reports. This is used to replace the teacher table in the new year.

The preregistration system has the following scripts in **cgi/prereg**:

- **asclass.pl** - assign student homerooms (preregistration) in prereg.
- **copybackstaff.pl** - copy prereg staff tables back into normal staff tables. Current staff values are removed.
- **copystaff.pl** - copy the staff (staff, staff_multi) tables into the prereg staff tables (prereg_staff, prereg_staff_multi).
- **premptyclasslist.pl** - older custom classlist report pointed at the preregistration student table. This should be replaced with the custom classlist report being able to accept table input.
- **preregempty.pl** - simple script to delete prereg student table (with confirmation).
- **preupdateroom.pl** - update the normal student table grade and homeroom values from the student preset table, which basically carries the homeroom and grade.
- **prexfer.pl** - selectively move preregistered students into the normal student table. (prereg to student).
- **promote.pl** - empty the preset table. Then add new records with incremented grades into preset. Also include students in prereg, without grade change.
- **prstudview.pl** - an old prereg (student) viewer. Should be updated or use new student viewer pointed at this table.
- **prstudview1.pl** - a preset table viewer to show homeroom and grade assigned.

The Staff Management System



This system includes the 2 tables above (staff, and staff_multi) linked with the common userid field (which also links the staff to other tables such as the subject table in the report card system, etc.

This system is supported by the meta system and both editing and reports are generated using templates.

The scripts, located in /cgi/staff, are used to add, edit, and report on staffs. The staff report is template based and can load multiple formats.

The staff management system has the following scripts in **cgi/staff**:

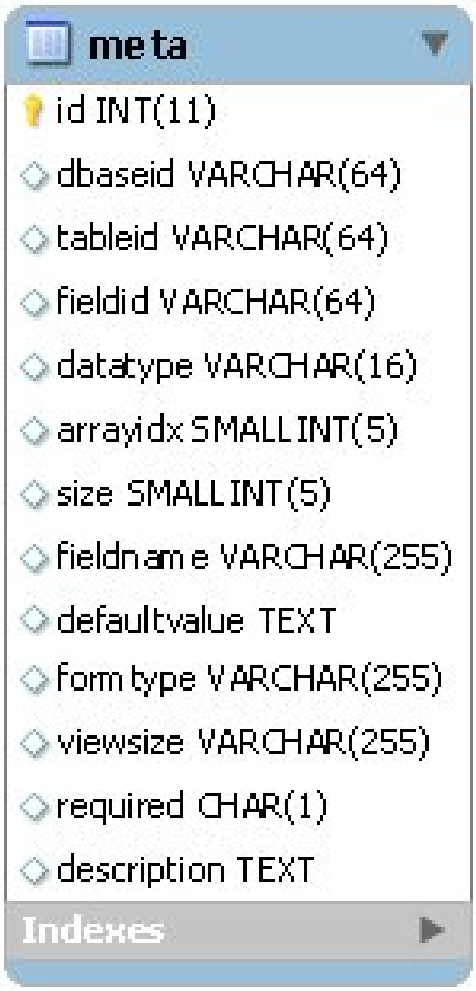
- **staffadd.pl** - Add new staff members.
- **staffdel.pl** - Delete existing staff members; called by staffdeled.pl
- **staffdeled.pl** - Edit/Delete Staff Members
- **staffed.pl** - Edit staff members; called by staffdeled.pl
- **staffrpt.pl** - Staff report, based on various templates; pdf output.
- **staffview.pl** - Simple staff viewer with most common fields.

The Date System

The **dates** table is used to store dates for the school year. They indicate days (Monday - Friday) that school is **not** in session due to statutory holidays, school holidays, inservice, etc. This is required to correctly calculate student attendance since OA only tracks days that students are away, not present.

These date scripts are located in /cgi/eoy and include:

- **dateadd.pl** - add dates to the school year.
- **dated.pl** - edit school dates.
- **datedel.pl** - delete school dates; called by datedeled.pl
- **datedelall.pl** - delete all school dates. (ie. school dates)
- **datedeled.pl** - edit/delete schoold dates.
- **dateview.pl** - view school dates (and other date configuration in conf files.



The metadata system allows us to provide metadata (data about data) about values in fields of tables. Currently this supports the student tables (student, studentwd, prereg), the staff tables (staff, staff_multi), and also the student_inac tables (used in Canada for First Nations schools).

The meta table provides information about each field in those tables including, text name (supporting translation), default values, type of edit form used (plus other edit characteristics) and whether a required field during entry.

There is a libmeta library function in the lib folder. It provides the metaInputField function to provide entry form values 'around' the staff or student fields (identified by the fieldid).

The entry method in general is:

1. Read the template into a text variable. Templates are stored in the templates folder at the same level as the cgi, etc, admin folders and have a filename .tpl extension.
2. Read in the field values for the table of interest from the meta table. This includes the fieldid and the fieldname (which may be a translated value updated by the translation

system). These are loaded into a hash to make it easy to lookup a fieldname based on a fieldid in the template.

3. Replace fieldid's in the template text (identified by <*field*> values where 'field' is the fieldid) with the fieldnames loaded from meta table.

4. Next load record values (if any) into a hash indexed also by fieldid (which is the name of the field in the table). Then use a regex expression to search the template for any fieldid of the form <@field@> and replace those values with the value from the staff or student table. This will allow us to display the names and values of a particular student or staff record.

5. If there is a desire to edit values, then use the metaInputField function in a form. Use the same search and replace method to return form elements (with embedded value) into the form text.

If this is a addition operation rather than an edit operation, there are no existing values. As a result, there is no need have a read values step and nothing is passed to metaInputField for the value.

Here is some example code from the inacadd.pl script for an edit operation. A display only operation does not require the metaInputField loop.

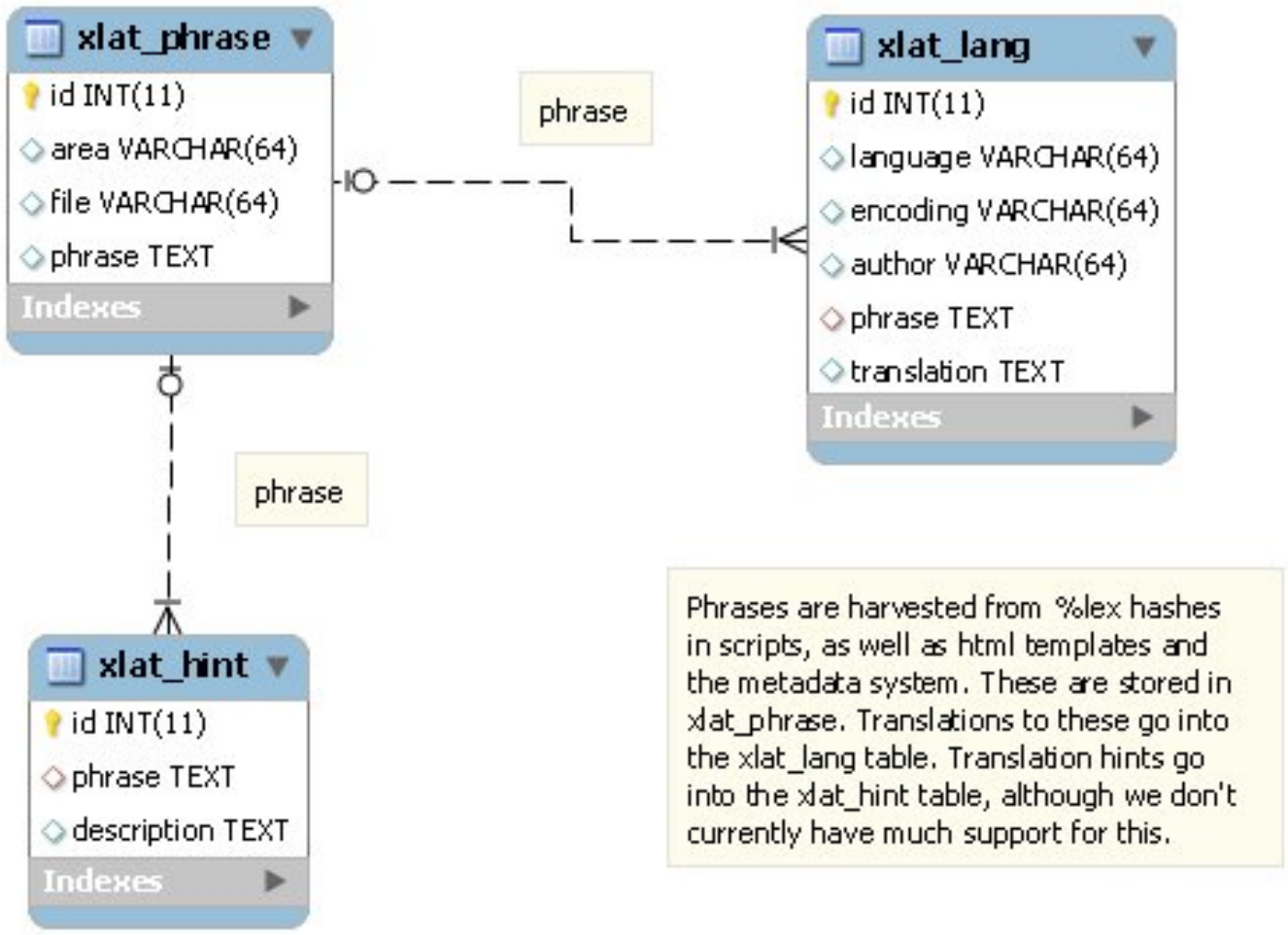
```
# Read in Template
unless (open (FH,''<../../template/inac.tpl'')) {
  print $lex{'Unable to open template file:'},''$!\n'';
  die $lex{'Unable to open template file:'},''$!\n'';
}
my $formtext;
{ local $/; $formtext = <FH>; close FH;}

# Get fieldnames from meta, store in hash.
my $sth = $dbh->prepare(''select fieldid, fieldname from meta where tableid
$sth->execute( 'student_inac' );
if ( $DBI::errstr ) { print $DBI::errstr; die $DBI::errstr; }
my %fieldnames = ();
while ( my ( $fieldid, $fieldname ) = $sth->fetchrow ) {
  $fieldnames{$fieldid} = $fieldname;
}

# Now replace <*fieldid*> with fieldnames; this gives us the text;

$formtext =~ s{\<\*(.*?)\*\>}
{ exists( $fieldnames{$1} )
     ? $fieldnames{$1}
     : $1
     }gsex;
```

```
# Formtext is now ready for multiple use; only contains <@fieldid@> values.
# now parse for form entry replacement elements  <@name@>
# Extract fields from template
my @fields = ();
while ( $formtext =~ m/\<\@(.*)\@\>/g ){
  push @fields, $1;
}

# Get their record, if any.
$sth->execute( $studnum );
if ( $DBI::errstr ) { print $DBI::errstr; die $DBI::errstr; }
my $recref = $sth->fetchrow_hashref;
my %fieldvals = %{ $recref };

# get replacement form element values for fields
foreach my $fieldid ( @fields ) {
    $values{$fieldid} = metaInputField('student_inac',
      $fieldid, $fieldvals{$fieldid}, $dbh, $studnum );
}

# now put form elements (including values) back into $formtext variable...
$formtext =~ s{ \<\@(.*?)\@\> }
   { exists($values{$1})
       ? $values{$1}
       : ''$values{$1}-$1''
   }gsex;
```

The metadata system has the following scripts in **cgi/meta**:

- **metaedit.pl** - edit metadata for student and staff.
- **metaupdate.pl** - synchronize meta table values with underlying data (ie. student, staff) tables.

The Translation System



The translation system has the following scripts in **cgi/xlat**:

- **fixCaseDup.pl** - finds phrases with duplication, differing only in case. Doesn't really fix anything, just displays. Not linked to translation on eoy page.
- **xlatRemoveDup.pl** - this script does remove duplicate phrases. Not linked to eoy page.
- **xlatExportPO.pl** - export phrases into a .PO file. The start page gives a choice of phrases and translation. These come out of the xlat_lang table.
- **xlatImportPO.pl** - upload a .po file containing phrase translations into the xlat_lang table (translation) and also xlat_phrase if required (with options).
- **xlatExtract.pl** - read scripts, html templates, and metadata (table) and generate a list of all phrases from the lex hashes. These are placed in the xlat_phrase table.
- **xlatInsert.pl** - will insert translations (from xlat_lang table) into the lex hashes in the scripts, rewrite the html pages from the html templates (with translation), and also update the meta table fieldname values.

- **xlatPhraseView.pl** - view phrases and translations from the xlat_lang and xlat_phrase table.
- **xlatTransEdit.pl** - Edit translation phrases.

External User Management System

The external user management system has the following scripts in **cgi/usermanage**:

- **addexternal.pl** - copy user information (staff and student) onto an external server. Cron scripts on the external server will compare this file with existing users and add any missing ones.
- **resetexternal.pl** - copy a reset file to the external server to change password or lock the account (or the reverse).
- **gdocview.pl** - View Google Users in your educational domain, and compares them to existing local records.
- **googleupdate.pl** - Manage Student and Staff accounts on Google domain.
- **googleview.pl** - Simpler Google Viewer than gdocview. Only shows name.

The LDAP scripts manage users on an external ldap server. The ldap area has the following scripts in **cgi/ldap**:

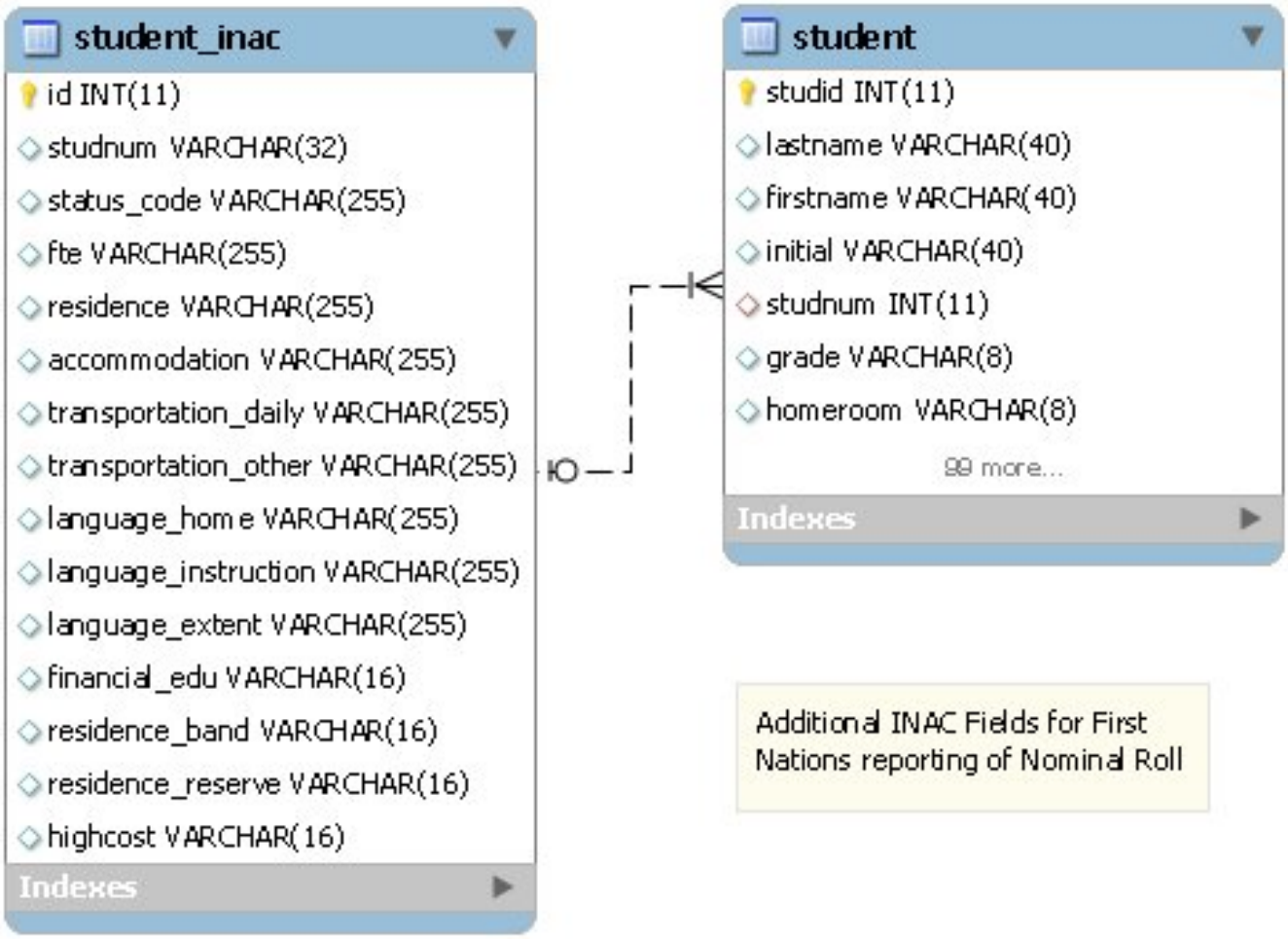
- **ldapmanage.pl** - fill missing ldap values in the OA student and staff tables, so this can then be synchronized with ldap directory. File upload is not yet supported, but in the cards.

  ldifexport - ldif export scripts (no longer required). old - old scripting directory. Saved, just in case...
- **resetldap.pl** - reset passwords and user accounts in the ldap directory.
- **syncstaff.pl** - synchronize OA staff and the ldap directory.
- **syncstudent.pl** - synchronize OA students and the ldap directory.
- **viewldap.pl** - view the ldap directory and values. Basically an ldap dump. Not linked to the main page, so a blind url.

The SIRS3 (another SIS) import system has the following scripts in **cgi/dbfimport**:

- **importstudentdbf.pl** - Import students from SIRS3 dbf upload.
- **importhomelogicpassword.pl** - Import the passwords (stored in homelogic module) of SIRS.

INAC System

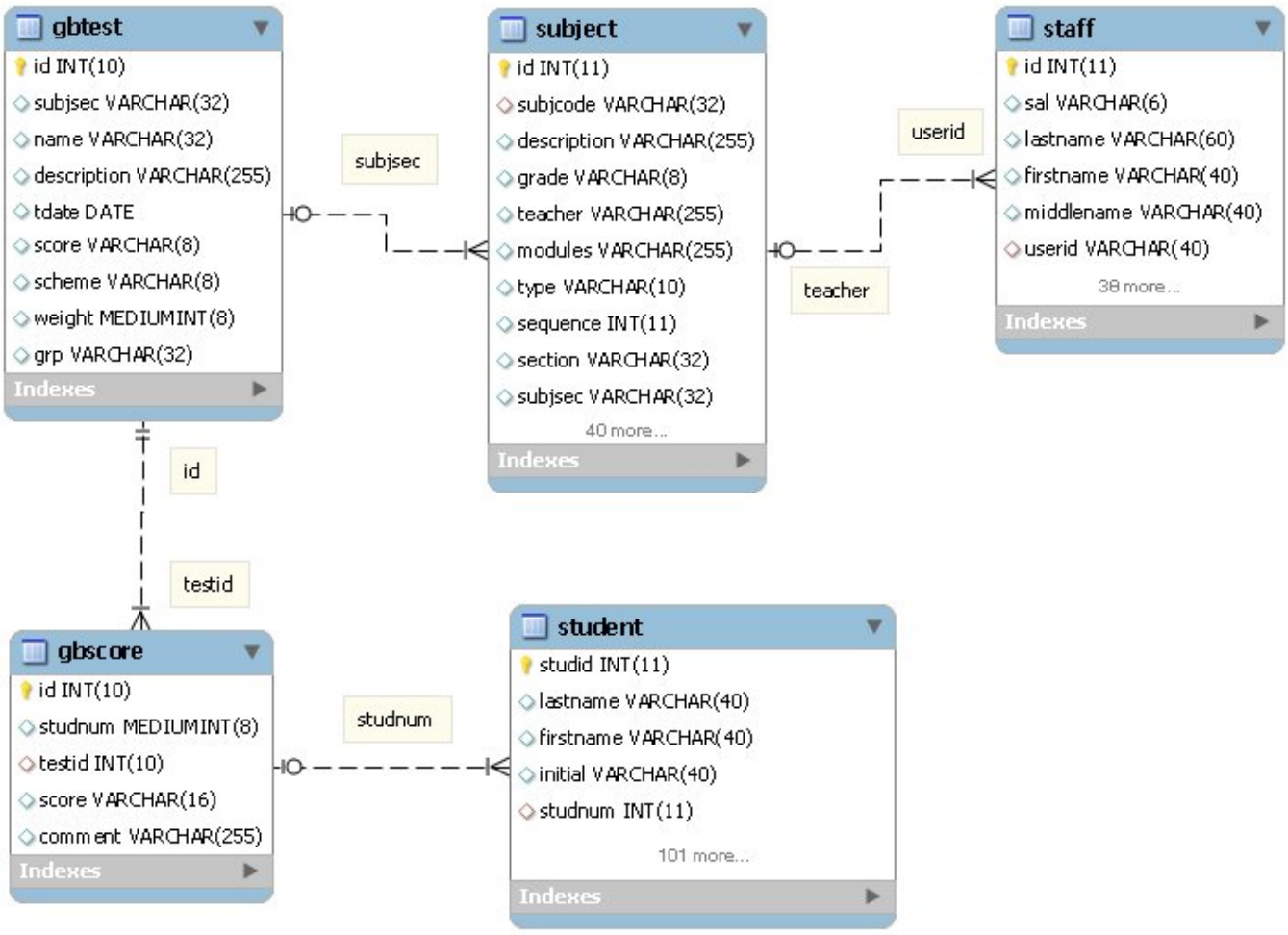


This is an additional table and scripts addon to the student demographics system. It provides the data required to print INAC reports such as the nominal roll.

The inac area has the following scripts in **cgi/inac**:

- **canada.jpg** - canadian flag image - inac report
- **fip-eng.jpg** - image for inac report
- **inacadd.pl** - add inac fields to the new student_inac table. Additional fields are required for the inac nominal roll forms.
- **inacdeled.pl** - normal edit/delete script for these values.
- **rptinacconf.pl** - inac confirmation report (html).
- **rptinacroll.pl** - inac nominal roll report (pdf).

The Gradebook



The Gradebook uses two tables:

1. **gbscore** - holds individual student scores for a particular test / assessment item.
2. **gbtest** - holds information about each test/assessment item for all subjects.

Password Policy

Configuration values are stored in the etc/admin.conf configuration file for each school. There are 2 groups; one for student passwords and one for staff passwords.

```
$g_studentpwd_minlen = 4;
$g_studentpwd_maxlen = 6;
$g_studentpwd_genlen = 6;
$g_studentpwd_signs = 0;
$g_studentpwd_caps = 0;

# Staff Password Generation
$g_staffpwd_minlen = 6;
$g_staffpwd_maxlen = 0; # unlimited
$g_staffpwd_genlen = 6;
$g_staffpwd_signs = 0;
$g_staffpwd_caps = 1;
```

The values affect both password generation and password checking.

The minimum length (minlen) is the minimum length, both for generating and checking. The maxlen is the maximum length a password can be for checking. The genlen is the length of the generated password. The signs are used for checking the presence of non-alphanumeric values in passwords. The caps is used for the generation and checking of capitals present in the password.

Two different perl modules are used for passwords:

1. **Generate Passwords** (Crypt::GeneratePassword) - is used to generate passwords (typically via the meta library function.
2. **Check Passwords** (Data::Password) - is used to check password quality when editing or storing password values.

Other Settings for Checking/ Generating passwords, that are found in scripts (and therefore are default behaviours) also include lang, minfreq, avgfreq, and the following:

1. $DICTIONARY - Minimal length for dictionary words that are not allowed to appear in the password. Set to false to disable dictionary check.
2. $FOLLOWING - Maximal length of characters in a row to allow if the same or following. If $FOLLOWING_KEYBOARD is true (default), the module will also check for alphabetical keys following, according to the English keyboard layout. Set $FOLLOWING to false to bypass this check.
3. $GROUPS - Groups of characters are lowercase letters, uppercase letters, digits and the rest of the allowed characters. Set $GROUPS to the number of minimal character groups a password is required to have. Setting to false or to 1 will bypass the check.

4. $MINLEN / $MAXLEN - Minimum and maximum length of a password. Both can be set to false.

IEP Data Overview

Personnel Identification

Teachers, Administrators and EA/TAs in the schools will be listed in a teacher table in the schools. As such they can be identified by a school:userid approach. For example a teacher in a school called St Mary's might be stmary:fredflintstone.

Other School Division staff not associated with a school, are stored in a single table listing their userid, name, agency (if not central office), and phone, email, and designation (psychologist, etc.)

Subjects and Evaluation

We have divided educational expectations into different *subjects*, some of which are traditional subjects; some are not.

The subject areas are given numbers (and associated text). For example, *Language - Receptive* is number 105 while *Gross Motor Development* is 200.

Within each subject area, particular objectives are given numbers to identify them. For example, within *105 Language - Receptive*, the first objective is *1 - Show Awareness of Speaker* and would have a unique identification of 105-1 while the the third, *3 - Look in response to own name*, would be 105-3.

When setting up an evaluation program for students, particular learning outcomes/objectives from each subject (105-1, 105-3) are added to a student's evaluation master for each subject. (S)he will then be evaluated throughout the year in each subject based on the objectives chosen for him/her.

Evaluation Workflow

Groups of objectives are added for each student in an evaluation master. We call these groups of objectives, *subjects*. Each of those *subjects* can have up to 32 objectives from that subject area.

When evaluations are done each term, a teacher will evaluate each student in each subject. If a subject record for this term already exists, its values are read. If no record exists for this term, a new one is created. Values and commentary are entered for this record and then it is stored.

Evaluations:

1. Choose it Enter Evaluations. All students are listed, with options to add evaluations to existing ones or edit existing ones (all listed).
2. If Add, a new blank record with values for subject displayed for entry. If Edit, an existing record may be changed.

Report cards will be generated for this student and all evaluation records for this and previous terms will be read and used to create a simple report card.

Table Structures

In general we will link to other information in the school's database for contact information, etc. although this is only loosely linked so the IEP system could very well be stand alone.

- **Special** - A table of student data.

  id - record id
  lastname - char(40) - student lastname
  firstname - char(40) - student firstname
  birthdate - date - birthdate
  studnum - integer - division student number
  provnum - varchar(12) - provincial/state student number
  desdate - date - date of designation
  ddpflvl - char(6), - ddpf level
  designation - char(60) - designation
  grade - char(6) - grade placement
  school - char(60) - school (Name and code)
  sex - char(1) - gender of student
  medical - text - description of medical condition.
  medication - text - medication required.
  equip - text - required equipment for student.
  adapt - text - school physical adaptations required by this student.

- **Personnel** - a table listing central office staff and other agency personnel not located in a school staff table.

  id - record number
  userid - a short field identifying this person.
  lastname
  firstname
  jobtitle
  agency
  phone
  email

- **Team** - a table listing the members of each team that work with and are responsible for student. Parents are considered to be nominal members. There will be one record for each team member for each student. The entire table will therefore hold all team listings for all students.

  id - record id.
  studnum - number of the student whose team this member is on.
  userid - userid and school/agency identifying this person.
  jobtitle - role
  phone - redundant contact info
  email - redundant contact info

- **Assess** - A table listing assessments done on students.

  id - record id
  studnum - student number
  category - char(255) - text matching category from assesscat.
  reqdate - date assessment requested.
  resdate - date assessment completed.
  comment - text - commentary on assessment.

- **CatAssess** - a table of assessment categories used in category field of assess field.

  id - record id
  category - char(255) - text specifying category

- **Subject** - a table containing subject categories and objectives for students.

  id - record id.

  catnum - integer - a number specifying the category of this objective.

  category - text - a word or phrase describing category of this objective - these two fields are tied.

  idnum - int - a number for this objective/expectation. Identifies it within the category.

  grade - char(8) - possibly the grade of the expectation (some subjects) - rarely used.

  description - description of the objective/expectation - tied to idnum. They go together.

  commentcodes - char(255) - codes linking to a comment table; unused currently.

- **EvalMst** - evaluation master record for students; one per student per subject.

  id - record id

  studnum - integer - the student number

  subnum - integer - the category number (catnum) from the subject table (The idnums and descriptions of the objectives from the subject table are stored in each of the individual objectives obj1, obj2, etc). There is no need for lookup into the subject table once evaluation masters are created at the start of the year.

  method - text - a brief description of the teaching techniques used.

responsible - text - who is responsible for implementing this subject.

obj1, obj2, ... obj32 - objectives/expectations. - text - contain text description of the subject.

- **Eval** - evaluation records for students; one per student per subject per term. Thus each student will have many eval records for each term of the year. For each subject there will be one evalmst record and up to 4 eval records (depending on the number of terms in the school).

  Note: First fields common with the evalmaster; only term and commentary extra.

  id - record id
  studnum integer - the student number
  subnum integer - matches subject in EvalMst table.
  obj1, obj2, ... obj32 - char(16) objective results
  term - the term number for this record.
  comments - text; a long text field for commentary

Schedule Development

In order to create a timetable we have to know certain values:

1. DaysPerCycle - The cycle that the days are built around. In some schools this might be 1 if all days are the same for a very short school term. If following a normal week it might be 5. A bi-weekly timetable would be 10. Our school operates on a 6 day cycle.

   This must be uniform across all grades in the school. If not, you'll have to run 2 copies of Open Admin, since they are basically different schools in the same building.

2. multiTerm - Different grade levels may have a different number of terms in the year with different start and ending dates. For example, grades K-6 might have 3 terms per year, while grades 7-12 might have 4. This can be accomodated within Open Admin by setting values in the main configuration file, etc/admin.conf, turning on multiTerm ($multiTerm = 1), and setting the dates for the different terms. Each grade is associated with one of these term tracks.

3. PeriodsPerDay - Each grade can have a different number of periods per day (ppd). Again, this is set in the etc/admin.conf file.

The schedat table consists of:

1. id - unique access key

2. day - a DayInCycle value (based on the number of days in the school cycle) such as 1 or 2 or 3.

3. period - the period number of the day (starting from 1). It may have a value up to the number of PeriodsPerDay for that grade.

4. subjsec - the subject-section value for a particular class.

5. term - one of the school terms. There are entries for each school term to make a, possibly unique, timetable for each term for each grade. (It may or may not change from term to term) A term is a period of time with tests/assessments at the end.

6. grade - a grouping of students based on age and ability level.

A **timetable block** will be all the records for a particular grade for a particular term. (ie. The Grade 7 timetable for Term 1.) Each cell in the timetable will have one or more subject-sections (in case of backings). Each cell will have a day (ie. DayInCycle), a period, and a subject-section (called subjsec in scripts) in order to describe it.