# Math System Database Documentation

NOTE: This document is maintained in the **mathops\_db** project, under the **docs** folder.

## Section 1. Structure

### Current Design

There is currently one database named "math", for which there exist both production and development instances. Both run on the "baer" server in Informix.

The production "math" database is intended to be taken down between each semester for final grading and semester roll-over processing. Some systems, like the Math Plan tool, can run from the development instance during this downtime.

There are also many archive databases, one for each term, with a snapshot of a subset of that term's production database at the end of the term. The structures of these databases are inconsistent since changes made after a term are not retroactively applied to earlier archives.

### Proposed Design

The proposed design would consist of a single PostgreSQL cluster with three tablespaces, three databases, and several schemas. Tablespaces are directories under which data is stored, and are separated to allow for segregation onto separate drives as needed for performance or capacity.

Databases are separated to support independent backup and recovery of each database.

Owning role:

* math

Tablespaces (owned by ‘math’) will include:

* primary\_ts /opt/pgsql17/pcschemas/primary all schemas except analytics and term
* analytics\_ts /opt/pgsql17/pcschemas/analytics the analytics schema
* term\_ts /opt/pgsql17/pcschemas/term all term schemas

Databases (owned by ‘math’) will include:

* math the production database (includes all schemas)
* math\_dev the development database (includes all schemas)
* math\_test a test database (includes all schemas, but only a generic “term” schema rather than one per term)

### Schemas

1. A **system** schema with tables related to the operation of the system, independent of any institution data. For example, user logins and role permissions, or system confitguration parameters.
2. A **main** schema with data that does not change from term to term (courses, students, transfer credit, math placement data, exam and homework definitions, and so on). This schema includes the definitions of terms. This schema will remain open during final grading and end of semester processing but may receive updates during the actual switch between semesters.
3. A schema for each term, with names of the form **fa25**, **sp25**, or **sm25**, that contain all data particular to a term, including course sections, student registrations, student exam and homework records, and so on. During each term, the next term’s data can be populated ahead of time and tested, and then when a term ends, the active term can be moved forward (in the main schema), and the new term schema will begin receiving data. Meanwhile, the schema for the ending term can be frozen against outside modification, and final grading performed, after which that schema is considered the archive of that semester's data. It may be desirable as well to archive parts of the "common" database each term to retain a snapshot of the configuration of courses and exams for that term.
4. An **extern** schema with data imported from external sources for fast local access. This data is never updated except by the process that imports from the external source.
5. An **analytics** schema with tables to support reporting and analytics, aggregating summary data over many terms or about courses outside Math. These tables may be very large but may be emptied and refreshed immediately before running a large analysis.
6. A **legacy** schema with the structrure of the old Informix database, used during the transition to the new schema. Tables can be slowly migrated from this schema into the new schemas and deactivated here until none remain.
7. A **live** schema that reflects the structure of external “live” data sources like Banner. This schema is not controlled by the MathOps system, and would change with each institutional customization.
8. An **ods** schema that reflects the structure of an external “operational data store” with near-live data optimized for query. This schema is not controlled by the MathOps system, and would change with each institutional customization.

Data for several future terms would always exist with some data relating to system maintenance windows, calendar dates, etc. This avoids storing data for several terms in confusing formats in the common database.

For Incomplete courses, there would be a record in the current term's database that would reference data in earlier terms (allowing the earlier term's deadlines and possible deadline overrides to be considered).

### Database Tables and Record Classes

Records (Java objects) should match database tables field for field, with no extra or missing fields. In the **legacy** schema, this may mean some fields are not used by the application. When migrating a data object from the legacy to another schema, a new table should be designed, a corresponding new record object should be created, and the old table and record should be abandoned once the conversion is complete.

This allows record objects to be organized by schema.

A generalized persistence layer could replace record classess with a single general record class interpreted through data-driven table definitions, with separate general classes for match criteria and updated data values.

### Caching

The database is the authoritative source of truth for all system data. Caches can support faster access, but data is not “official” until committed to the database, and in the event of conflict, the database is considered correct.

There will be a single database server (baer), but multiple client machines connecting to that server: the production web server (numan), the development web server (havoc), the video and media web server (nibbler), testing center checkin and checkout systems, and any system running the administrative application.

Due to this, long-term caching of data is not useful unless the cache itself is on the single database server and is used instead of direct database access for all updates or queries to an object. This architecture has other advantages – it could potentially allow the database engine to be paused, restarted, or even upgraded, while allowing the system to run entirely against the cache, queuing updates during the downtime, and draining that queue into the database once it returns to service. It would not, however, permit reboots of the database server without interruption of service unless the cache could be relocated on demand to a secondary server (much more complex).

However, caching is still useful on a single web server, since data may be used many times during a single page request. To that end, a per-request cache object can be constructed that can perform stock queries and cache results, with the understanding that storage will be short-lived. The cache object can implement common query logic, and may query blocks of data then return filtered results without additional queries.

If we refer to caching in this document, we will refer to the long-term central cache as “Level 1 Cache”, and the short-term per-request cache on individual servers as a “Level 2 Cache”. The Level 1 Cache would be considered by web services as authoritative, since they would not access the database directly.

### Java Package Structure

All database code lives in the **mathops\_db** project, under the **dev.mathops.db** top-level package.

Sub-packages include:

* **.cfg** Management of database configuration to allow clients to connect.
* **.field** Data types for field values.
* **.logic** Business logic (does this belong here?)
* **.schema** The various schemas in the system. Sub-packages include:
  + **.system** The system schema. Sub-packages for all schemas include:
    - **.rec** Record classes.
    - **.impl** Implementations of JDBC operations.
  + **.main** The main schema
  + **.term** The term schema
  + **.extern** The external data schema
  + **.analytics** The analytics schema
  + **.legacy** The legacy schema
  + **.live** The live schema
  + **.ods** The ODS schema

### Obsolete tables in Legacy database:

* **(?) applicant** (used to be used to hold applicant records from ODS while they were processed into student records)
* **bogus\_mapping** (used to be used for bogus sections to track student-reported registrations until verified)
* **ddcode** (code documentation)
* **dddomain** (domain documentation)
* **ddelement** (data dictionary documentation)
* **ddelement\_report** (data dictionary elements used in reports)
* **ddelement\_screen** (data dictionary elements used in screens)
* **ddreport** (report documentation)
* **ddscreen** (screen documentation)
* **ddtable** (table documentation)
* **ddtable\_element** (table element documentation)
* **delphi** (used to be used to import course registration records from ODS)
* **delphi\_check** (used to be used to import course registration records from ODS)
* **examqa** (used to store the answer key for each version of exams)
* **fcr** (a synonym for final\_croll)
* **fcr\_student** (used to be used when importing and reconciling final class roll from ODS)
* **fcrstu** (a synonym for fcr\_student)
* **(?) final\_croll** (used to be used when importing and reconciling final class roll from ODS)
* **(?) grade\_roll** (used to be used when importing and reconciling final class roll from ODS)
* **index\_descriptions** (documentation of indexes)
* **index\_frequency** (used by batch job to reindex database)
* **mdstudent** (used to be used to track Math Day student performance)
* **next\_campus\_calendar** (used to be used to preload next-term data)
* **next\_csection** (used to be used to preload next-term data)
* **next\_milestone** (used to be used to preload next-term data)
* **next\_remote\_mpe** (used to be used to preload next-term data)
* **next\_semester\_calendar** (used to be used to preload next-term data)
* **prev\_milestone\_appeal** (stores milestone appeals from the prior term)
* **prev\_extensions** (stores pace appeals from the prior term)
* **prev\_stlmiss** (used to be used to store prior term missed deadlines)
* **prev\_stlock** (used to be used to store prior term lockout records)
* **prev\_stmilestone** (used to be used to store prior term student milestone overrides)
* **prev\_stterm** (used to be used to store prior term student term records)
* **stc** (used to be used to store student course data temporarily)
* **stmdscores** (used to be used to store Math Day scores for students)
* **stuid\_tables** (I believe this was used during the switch from SSN to CSU ID)
* **sysmenus** (names and titles of system menus)
* **sysmenuitem** (menu items)
* **tree\_path** (tree structure storage)

### Organization by “Data Service”

**Term Service**

**Term** [**main**] - defines all terms (**term**)

**TermWeek** [**term**] - defined weeks of each term (**semester\_calendar/term\_week**)

**PaceTrackRule** [**term**] - rules to classify students into tracks (**pace\_track\_rule**)

**RuleSet** [**term**] - rules under which a section can operate (**pacing\_structure/rule\_set**)

**RuleSetRule** [**term**] - rules within a rule set (**pacing\_rules/rule\_set\_rule**)

**Student Service**

**Student** [**main**] - all students for whom we have current or historic data (**student**)

**StudentAccommodation** [**main**] - accommodations for students (**pace\_appeals**)

**StudentCategory** [**main**] - special categories to which the student belongs (**special\_stus**)

**StudentDiscipline** [**main**] - disciplinary incidents for each student (**discipline**)

**StudentHold** [**main**] - holds on each student account (**admin\_hold/stu\_hold**)

**StudentVisit** [**main**] - tracking student visits to the facility or to individual areas (**stvisit**)

**NewStudent** [**term**] – a student who is new this semester, for high school reporting (**newstu**)

**LiveAdvisee** [**ods**] - A student / advisor relationship.

**LiveStudent** [**live**] - a student record

**Profile Service**

**Profile** [**main**] - a container for a set of profile questions (**surveyqa**)

**ProfileQuestion** [**main**] - a question within a profile (**surveyqa**)

**ProfileQuestionChoice** [**main**] - a choice for a question within a profile (**surveyqa**)

**StudentProfileResponse** [**main**] - student response to a profile question (**stsurveyqa**)

**Course Service**

[**CatalogCourse**](#_2.2_CatalogCourse_[common]) [**main**] - a course listed in the university catalog

[**CatalogCoursePrereq**](#_2.3_CatalogCoursePrereqTrack_[commo) [**main**] – prerequisites for a course

[**Course**](#_2.6_Course_[common]) [**main**] - definition of each course (including placement, tutorials) (**course**)

**CourseSection** [**term**] - sections offered each term (**csection**)

**CourseSectionUnit** [**term**] - per-unit data for sections (**cusection**/**crsection**)

**CourseUnit** [**term**] - units of each course (**cunit**)

**CourseUnitObjective** [**term**] - objectives of each course unit (**cuobjective**)

**GradingStandard** [**term**] – rules for how grades are calculated (**grading\_std**)

**Lesson** [**main**] - lessons that make up courses

**LessonComponent** [**main**] - components that make up lessons

**UngradedSection** [**term**] - section whose grades will not be submitted (**dont\_submit**)

**Prereq** [**main**] - course prerequisite relationships (**prereq**)

**Registration Service**

**StudentCourse** [**term**] - registrations for each student in the term (**stcourse**)

**StudentCourseUnit** [**term**] - student status in a unit of a course (**stcunit**)

**StudentLesson** [**term**] - state of student work in objective (**stcuobjective**)

**StudentCourseMastery** [**term**] - student status in standards based course (**stu\_course\_mastery**)

**StudentUnitMastery** [**term**] - student status in unit of standards based course (**stu\_unit\_mastery**)

**DuplicateRegistration** [**term**] – record of students who have duplicate registrations (**dup\_registr**)

**GuestStudent** [**term**] – a student with guest access to a course (**except\_stu**)

**StudentIncomplete** [**term**] - incompletes from a prior term being finished

**StudentTransferCredit** [**main**] - transfer credit each student has (**ffr\_trns**)

**RequiredPrereq** [**main**] - the prerequisite needed to register for a course

**LiveCsuCredit** [**ods**] - Course credit earned by a student at CSU

**LiveReg** [**live**] - a student registration in a course

**LiveTransferCredit** [**ods**] - a transferred course credit from another institution

[**Cohort**](#_2.5_Cohort_[term]) [**term**] – cohorts to which students are assigned (**cohort**)

**Resource Service**

**Resource** [**main**] - lendable resources (**resource**)

**TestingCalculator** [**main**] - calculators lent to students in testing center (**calcs**)

**StudentResourceLoan** [**main**] - resources lent to student (**stresource**)

**Schedule Service**

[**Calendar**](#_2.1_Calendar_[term]) [**term**] - hours of center facilities, holidays (**campus\_calendar**)

**StandardMilestone** [**term**] - standard milestone dates in courses (**std\_milestone**)

**Milestone** [**term**] - milestone dates in courses (**milestone**)

**StudentStandardMilestone** [**term**] - deadline overrides for students (**stu\_std\_milestone**)

**StudentMilestone** [**term**] - deadline overrides for students (**stmilestone**)

**StudentPaceSummary** [**main**] – historical summary dta (**stpace\_summary**)

**StudentTerm** [**term**] - student summary status for the term (**stterm**)

**StudentAppeal** [**term**] - documentation of student deadline appeals (**milestone\_appeal**)

**Placement Service**

[**ChallengeFee**](#_2.4_ChallengeFee_[common]) [**main**] - fees charged for challenge exams (**challenge\_fee**)

**PlacementExam** [**main**] – placement exam versions (**mpe**)

**PlacementFee** [**main**] - fees charged for placement exams (**plc\_fee**)

**PlacementLog** [**main**] - log of placement attempts (**mpe\_log**)

**PlacementScoreQueue** [**main**] – placement results to be sent to Banner (**mpscorequeue**)

**UnproctoredPlacement** [**term**] - when unproctored placement available (**remote\_mpe**)

**StudentPlacementAttempt** [**main**] - student placement attempts (**stmpe**)

**StudentPlacementAttemptAnswer** [**main**] - a placement question answer (**stmpeqa**)

**StudentPlacementCredit** [**main**] - placement credit earned by a student (**mpe\_credit**)

**StudentPlacementDenied** [**main**] - placement credit denied to a student (**mpecr\_denied**)

**StudentChallengeAttempt** [**main**] - student challenge exam attempts (**stchallenge**)

**StudentChallengeAttemptAnswer** [**main**] - a challenge exam question answer (**stchallengeqa**)

**StudentMathPlan** [**main**] – math plan responses for each student (**stmathplan**)

**Assessment Service**

**Assignment** [**main**] - definitions of assignments (**homework/assignment**)

**MasteryExam** [**main**] - definitions of mastery exams (**mastery\_exam**)

**Exam** [**main**] - definitions of exams (**exam**)

**StudentExam** [**term**] - course/tutorial exams taken by student in term (**stexam**)

**StudentExamAnswer** [**term**] - individual question results for student exams (**stqa**)

**StudentHomework** [**term**]- course homework submitted by students in term (**sthomework**)

**StudentHomeworkAnswer** [**term**] - individual question results for student homework (**sthwqa**)

**StudentLicensingAttempt** [**term**] - user's exam attempts for each student (**users**)

**MasteryAttempt** [**term**] - mastery exams taken by student in term (**mastery\_attempt**)

**MasteryAttemptQA** [**term**] - answers on mastery exam attempts (**mastery\_attempt\_qa**)

**EText Service**

**EText** [**main**] - definitions of e-text books (**etext**)

**ETextCourse** [**main**] - courses each e-text grants access to (**etext\_course**)

**ETextKey** [**main**] - activation keys for e-texts (available and used) (**etext\_key**)

**StudentEText** [**main**] - e-texts owned by each student (**stetext**)

**Messaging Service**

**Message** [**term**] - stock messages that can be sent to students (**msg**)

**StudentMessage** [**term**] - messages sent to students each term (**stmsg**)

**Testing Service**

**TestingCenter** [**main**] - defined testing centers (**testing\_centers**)

**TestingStation** [**main**] - configured testing stations (**client\_pc**)

**PendingExam** [**main**] - record of all open proctored exams (**pending\_exam**)

**System Service:**

**DbInfo** [**all**] - descriptor of the database being accessed (**which\_db**)

**LocalLogin** [**system**] - a login that can be used in place of an eID (**logins**)

**Parameters** [**system**] – parameters for batch jobs and reports (**parameters**)

**UserPermission** [**system**] – administrative permissions for each named user (**user\_clearance**)

**RepoprtPerms** [**system**] – administrative permissions for each named user (**report\_perms**)

**Lookup Service:**

**HighSchool** [**main**] - high schools for annual reporting (**high\_schools**)

**HoldType** [**main**] - defines holds that can be applied to student accounts (**hold\_type**)

**MessageLookup** [**main**] – table of text messages for use by other tables (**msg\_lookup**)

**ZipCode** [**main**]- zip codes (**zip\_code**)

**External Data Service:**

**…**

**Analytics Service:**

**TableName** [**analytics**] - description

## Section 2. Data Objects

Data objects are immutable representations of one object stored in the database (usually, but not always, as a single database record).

Within each object, fields that make up the *primary key* (those whose values uniquely identify an object) are **underlined**.

### 2.1 AssignmentRec [main]

Each object represents an assignment that can be used in a course or tutorial.

#### 2.1.1 Member Fields

**assignmentId** String The assignment ID.

**assignmentType** EAssignmentType The assignment type.

HW = HOMEWORK (used in old courses)

ST = STANDARD (used in standards-based courses)

EX = EXPLORATION (used in standards-based courses)

**courseId** String The course ID.

**unit** Integer The unit.

**objective** Integer The objective.

**treeRef** String The tree reference.

**title** String The assignment title.

**whenActive** LocalDateTime The date/time the assignment was made active.

**whenPulled** LocalDateTime The date/time the assignment was pulled from service.

#### 2.1.2 Backing Database Table

Each record in the backing table represents a single **AssignmentRec** data object.

**Informix table name: math.homework**

**PostgreSQL table name: main.assignment**

**PostgreSQL fields:**

**assignment\_id** varchar(20) The **assignmentId** field (Primary key).

**assignment\_type** char(2) The **assignmentType** field.

**course\_id** char(10) The **courseId** field.

**unit** smallint The **unit** field.

**objective** smallint The **objective** field.

**tree\_ref** varchar(250) The **treeRef** field.

**title** varchar(60) The **title** field.

**when\_active** timestamp(0) The **whenActive** field.

**when\_pulled** timestamp(0) The **whenPulled** field.

#### 2.1.3 Data Retention

Table data would remain in place indefinitely, being updated from time to time as assignments are created or retired.

#### 2.1.4 API

The API is exposed via the **AssignmentRecLogic** utility class.

boolean insert(Cache cache, **AssignmentRec** record);

boolean delete(Cache cache, **AssignmentRec** record);  
 List<**AssignmentRec**> queryAll(Cache cache);  
 List<**AssignmentRec**> queryActiveByCourse(Cache cache, String courseId);

List<**AssignmentRec**> queryActiveByCourseUnit(Cache cache, String courseId, Integer unit);

List<**AssignmentRec**> queryActiveByCourseUnitObjective(Cache cache, String courseId,Integer unit,

Integer objective);

**AssignmentRec** queryActive(Cache cache, String course, Integer unit, Integer objective,  
 String assignmentType);

**AssignmentRec** query(Cache cache, String assignmentId);

### 2.2 MasteryExamRec [main]

Each object represents a mastery exam that can be used in a standards-based course.

#### 2.2.1 Member Fields

**examId** String The exam ID.

**examType** EExamType The exam type.

ST = Standard mastery

**courseId** String The course ID.

**unit** Integer The unit.

**objective** Integer The objective.

**treeRef** String The tree reference.

**title** String The assignment title.

**buttonLabel** String The label for the button to take the exam.

**whenActive** LocalDateTime The date/time the exam was made active.

**whenPulled** LocalDateTime The date/time the exam was pulled from service.

#### 2.2.2 Backing Database Table

Each record in the backing table represents a single **MasteryExamRec** data object.

**Informix table name: math.mastery\_exam**

**PostgreSQL table name: main.mastery\_exam**

**PostgreSQL fields:**

**exam\_id** varchar(20) The **examId** field (Primary key).

**exam\_type** char(2) The **examType** field.

**course\_id** char(10) The **assignmentId** field.

**unit** smallint The **unit** field.

**objective** smallint The **objective** field.

**tree\_ref** varchar(250) The **treeRef** field.

**title** varchar(60) The **title** field.

**button\_label** varchar(50) The **buttonLabel** field.

**when\_active** timestamp(0) The **whenActive** field.

**when\_pulled** timestamp(0) The **whenPulled** field.

#### 2.2.3 Data Retention

Table data would remain in place indefinitely, being updated from time to time as assignments are created or retired.

#### 2.2.4 API

The API is exposed via the **MasteryExamRecLogic** utility class.

boolean insert(Cache cache, **MasteryExamRec** record);

boolean delete(Cache cache, **MasteryExamRec** record);  
 List<**MasteryExamRec**> queryAll(Cache cache);  
 List<**MasteryExamRec**> queryActiveByCourse(Cache cache, String courseId);

List<**MasteryExamRec**> queryActiveByCourseUnit(Cache cache, String courseId, Integer unit);

List<**MasteryExamRec**> queryActiveByCourseUnitObjective(Cache cache, String courseId, Integer unit,  
 Integer objective);

**MasteryExamRec** queryActive(Cache cache, String courseId, Integer unit, Integer objective,  
 String examType);

**MasteryExamRec** query(Cache cache, String examId);

### 2.3 MasteryAttemptRec [term]

Each object represents a student attempt on a mastery exam.

#### 2.3.1 Member Fields

**serialNbr** Integer The serial number of the exam session (one session may include multiple  
 mastery exam attempts, so this field value is not unique).

**examId** String The exam ID.

**stuId** String The ID of the student.

**whenStarted** LocalDateTime The date/time the exam was started.

**whenFinished** LocalDateTime The date/time the exam was finished.

**examScore** Integer The score the student achieved on the exam.

**masteryScore** Integer The score that was required to demonstrate mastery.

**passed** String The result of the exam.

"Y" if passed

"N" if not

"G" to ignore

"P" if taken away

**isFirstPassed** String "Y" if the first passed attempt for student on exam

**examSource** String The source of the exam.  
 "RM" if remote

"TC" if from a testing center

"HG" if hand-graded

#### 2.3.2 Backing Database Table

Each record in the backing table represents a single **MasteryExamRec** data object.

**Informix table name: math.mastery\_attempt**

**PostgreSQL table name: [term].mastery\_attempt**

**PostgreSQL fields:**

**serial\_nbr** integer The **serialNbr** field (Primary key).

**exam\_id** varchar(20) The **examId** field (Primary key).

**stu\_id** char(9) The **stuId** field.

**when\_started** timestamp(0) The **whenStarted** field.

**when\_finished** timestamp(0) The **whenFinished** field.

**exam\_score** smallint The **examScore** field.

**mastery\_score** smallint The **masteryScore** field.

**passed** char(1) The **passed** field.

**is\_first\_passed** char(1) The **isFirstPassed** field.

**exam\_source** char(2) The **examSource** field.

#### 2.3.3 Data Retention

Data is accumulated over each term, and remains in that term's schema as an archive.

#### 2.3.4 API

The API is exposed via the **MasteryAttemptRecLogic** utility class.

boolean insert(Cache cache, **MasteryAttemptRec** record);

boolean delete(Cache cache, **MasteryAttemptRec** record);

boolean updateResult(Cache cache, **MasteryAttemptRec** record, Integer newExamScore,  
 Integer newMasteryScore, String newPassed, String newIsFirstPassed);  
 List<**MasteryAttemptRec**> queryAll(Cache cache);  
 List<**MasteryAttemptRec**> queryByStudent(Cache cache, String stuId);  
 List<**MasteryAttemptRec**> queryByExam(Cache cache, String examId);

List<**MasteryAttemptRec**> queryByStudentExam(Cache cache, String stuId, String examId,  
 boolean passedOnly);

**MasteryAttemptRec** query(Cache cache, Integer serialNbr, String examId);

### 2.4 MasteryAttemptQaRec [term]

Each object represents a single question's answer on a student attempt on a mastery exam.

#### 2.4.1 Member Fields

**serialNbr** Integer The serial number of the exam session (one session may include multiple  
 mastery exam attempts, so this field value is not unique).

**examId** String The exam ID.

**questionNbr** Integer The question number.

**correct** String The outcome

"Y" if correct

"N" if not

#### 2.4.2 Backing Database Table

Each record in the backing table represents a single **MasteryAttemptQaRec** data object.

**Informix table name: math.mastery\_attempt\_qa**

**PostgreSQL table name: [term].mastery\_attempt\_qa**

**PostgreSQL fields:**

**serial\_nbr** integer The **serialNbr** field (Primary key).

**exam\_id** varchar(20) The **examId** field (Primary key).

**question\_nbr** smallint The **questionNbr** field (Primary key).

**correct** char(1) The **correct** field.

#### 2.4.3 Data Retention

Data is accumulated over each term, and remains in that term's schema as an archive.

#### 2.4.4 API

The API is exposed via the **MasteryAttemptQaRecLogic** utility class.

boolean insert(Cache cache, **MasteryAttemptQaRec** record);

boolean delete(Cache cache, **MasteryAttemptQaRec** record);

boolean updateCorrect(Cache cache, **MasteryAttemptQaRec** record, String newCorrect);  
 List<**MasteryAttemptQaRec**> queryAll(Cache cache);  
 List<**MasteryAttemptQaRec**> queryByAttempt(Cache cache, Integer serialNbr, String examId);  
 **MasteryAttemptQaRec** query(Cache cache, Integer serialNbr, String examId, Integer questionNbr);

### 2.5 StudentCourseMasteryRec [term]

Each object represents the mastery status of a student in a standards-based course. There will be a single record for each student registration in a standards-based course, which tracks the student's status with respect to eligibility for and mastery of course standards, as well as status with respect to exploration assignments. This record is sufficient to calculate a student's current score and grade without having to query all exam/assignment records for the student.

#### 2.5.1 Member Fields

**stuId** String The student ID.

**courseId** String The course ID.

**score** Integer The current score for the course

the sum of all score fields in StudentUnitMasteryRec objects

+ points for mastery/attempts of explorations (see below)

**nbrMasteredH1** Integer The number of standards the student has mastered in the first half.

**nbrMasteredH2** Integer The number of standards the student has mastered in the second half.

**nbrEligible** Integer The number of standards for which the student is eligible but has not

yet mastered (if this is 7 or more, new units may not be opened).

**explor1Status** String The student's status with respect to Exploration 1 (null until attempted).

'M' if assignment was mastered on time (worth 10 points)

'M1' if assignment was mastered 1 day late (worth 9 points)

'ML' if assignment was mastered late (worth 8 points)

'A' if assignment was attempted on time (worth 5 points)

'AL' if assignment was attempted late (worth 4 points)

**explor2Status** String The student's status with respect to Exploration 2.

#### 2.5.2 Backing Database Table

Each record in the backing table represents a single **StudentCourseMasteryRec** data object.

**Informix table name: math.stu\_course\_mastery**

**PostgreSQL table name: [term].stu\_course\_mastery**

**PostgreSQL fields:**

**stu\_id** char(9) The **stuId** field (Primary key).

**course\_id** char(10) The **courseId** field (Primary key).

**score** smallint The **score** field.

**nbr\_mastered\_h1** smallint The **nbrMasteredH1** field.

**nbr\_mastered\_h2** smallint The **nbrMasteredH2** field.

**nbr\_eligible** smallint The **nbrEligible** field.

**explor\_1\_status** char(2) The **explor1Status** field.

**explor\_2\_status** char(2) The **explor2Status** field.

#### 2.5.3 Data Retention

Records are created when a student starts a standards-based course and updated as the student makes progress within that course. Data is retained as a record within the term schema. In the Informix schema, this is cleared at the end of each term.

#### 2.5.4 API

The API is exposed via the **StudentCourseMasteryRecLogic** utility class.

boolean insert(Cache cache, **StudentCourseMasteryRec** record);

boolean delete(Cache cache, **StudentCourseMasteryRec** record);

boolean updateMastery(Cache cache, **StudentCourseMasteryRec** record, Integer newNbrMasteredH1,

Integer newNbrMasteredH2, Integer newNbrEligible)

boolean updateExplorations(Cache cache, **StudentCourseMasteryRec** record, String newExplor1Status,

String newExplor2Status);

boolean updateScore(Cache cache, **StudentCourseMasteryRec** record, Integer newScore);

List<**StudentCourseMasteryRec**> queryAll(Cache cache);

List<**StudentCourseMasteryRec**> queryByStudent(Cache cache, String stuId);  
 **StudentCourseMasteryRec** query(Cache cache, String stuId, String courseId);

### 2.6 StudentUnitMasteryRec [term]

Each object represents the mastery status of a student in a single unit of a standards-based course. There will be a set of 10 records for each student registration in a standards-based course, which tracks the student's status with respect to eligibility for and mastery of course standards. This record is sufficient to calculate a student's current score and grade without having to query all exam/assignment records for the student.

#### 2.6.1 Member Fields

**stuId** String[9] The student ID.

**courseId** String[10] The course ID.

**unit** Integer The unit (1 through 10).

**score** Integer The current score for the unit (0-15)

**srStatus** String[2] The student's status on the Skills Review (null until attempted).

'#' if attempted, the number is the number of questions correct so far

'P' if assignment has been passed

**s1Status** String[3] The student's status in Standard 1 (null until eligible)

'E' if student is eligible to demonstrate mastery

'ANM' if attempted but not mastered ("N" is the number of times  
 question 1 answered correctly, "M" is the number of times question 2  
 answered correctly, Example: "A00" if neither was correct.)

'M' if mastered on time (worth 5 points)

'ML' if mastered late (worth 4 points)

**s2Status** String[3] The student's status in Standard 2 (see above).

**s3Status** String[3] The student's status in Standard 3 (see above).

#### 2.6.2 Backing Database Table

Each record in the backing table represents a single **StudentCourseMasteryRec** data object.

**Informix table name: math.stu\_unit\_mastery**

**PostgreSQL table name: [term].stu\_unit\_mastery**

**PostgreSQL fields:**

**stu\_id** char(9) The **stuId** field (Primary key).

**course\_id** char(10) The **courseId** field (Primary key).

**unit** smallint The **unit** field (Primary key).

**score** smallint The **score** field.

**sr\_status** char(2) The **srStatus** field.

**s1\_status** char(5) The **s1Status** field.

**s2\_status** char(5) The **s1Status** field.

**s3\_status** char(5) The **s1Status** field.

#### 2.6.3 Data Retention

Records are created when a student starts a standards-based course and updated as the student makes progress within that course. Data is retained as a record within the term schema. In the Informix schema, this is cleared at the end of each term.

#### 2.6.4 API

The API is exposed via the **StudentUnitMasteryRecLogic** utility class.

boolean insert(Cache cache, **StudentUnitMasteryRec** record);

boolean delete(Cache cache, **StudentUnitMasteryRec** record);

boolean updateScore(Cache cache, **StudentUnitMasteryRec** record, Integer newScore);

boolean updateSrStatus(Cache cache, **StudentUnitMasteryRec** record, String newSrStatus);

boolean updateS1Status(Cache cache, **StudentUnitMasteryRec** record, String newS1Status);

boolean updateS2Status(Cache cache, **StudentUnitMasteryRec** record, String newS2Status);

boolean updateS3Status(Cache cache, **StudentUnitMasteryRec** record, String newS3Status);

List<**StudentUnitMasteryRec**> queryAll(Cache cache);

List<**StudentUnitMasteryRec**> queryByStudent(Cache cache, String stuId);

List<**StudentUnitMasteryRec**> queryByStudentCourse(Cache cache, String stuId, String courseId);  
 **StudentUnitMasteryRec** query(Cache cache, String stuId, String courseId, Integer unit);

### 2.7 TermRec [main]

Each object represents the definition of a term.

#### 2.7.1 Member Fields

**termKey** TermKey The term key.

**startDate** LocalDate The term start date.

**endDate** LocalDate The term end date.

**academicYear** String The academic year, like "1920"

**activeIndex** Integer The active term index (0 = current, 1 = next, -1 = prior, etc).

**dropDeadline** LocalDate The drop deadline.

**withdrawDeadline** LocalDate The withdrawal deadline.

#### 2.7.2 Backing Database Table

Each record in the backing table represents a single **TermRec** data object.

**Informix table name: math.term**

**PostgreSQL table name: [main].term**

**PostgreSQL fields:**

**term** smallint The **termKey** field's numeric form, like 202390 (Primary key).

**start\_date** date The **startDate** field.

**end\_date** date The end**Date** field.

**academic\_year** char(4) The **academicYear** field, like "2324".

**active\_index** smallint The **activeIndex** field.

**drop\_deadline** date The **dropDeadline** field.

**withdraw\_deadline** date The **withdrawDeadline** field.

#### 2.7.3 Data Retention

Records are created in the main schema for all existing terms, and as each new term starts, a new record is created. During the semester roll-over, the "active\_index" fields of all records are updated to indicate which term is currently active, and what relative position to the active term each term has (active = 0, prior = -1, next=+1, etc.) The currently active term is used to select which term schema should be used for current-term data.

#### 2.7.4 API

The API is exposed via the **TermRecLogic** utility class.

boolean insert(Cache cache, **TermRec** record);

boolean delete(Cache cache, **TermRec** record);

List<**TermRec**> queryAll(Cache cache);  
 **TermRec** queryByIndex(Cache cache, int termIndex);

List<**TermRec**> getFutureTerms(Cache cache);  
 **TermRec** queryActive(Cache cache);  
 **TermRec** queryNext(Cache cache);  
 **TermRec** queryPrior(Cache cache);  
 **TermRec** query(Cache cache, TermKey termKey);

### 2.8 StandardMilestoneRec [term]

Each object represents a milestone date for a standard within a course with a given pace index for a student with a given pace and pace track.

#### 2.8.1 Member Fields

**paceTrack** String The pace track for which the milestone applies.

**pace** Integer The pace for which the milestone applies.

**paceIndex** Integer The pace index for which the milestone applies.

**unit** Integer The unit for which the milestone applies.

**objective** Integer The objective for which the milestone applies (0 matches all in unit)

**msType** String The milestone type.  
 OP = Date objective opens to students (units 1-10, objective 1)  
 MA = Due date for mastery of standard (units 1-10)  
 EX = Due date for exploration (unit 11, objectives 1 and 2)  
 E1 = "One day late" date for exploration (as for explorations)  
 CD = Course deadline (unit 11, objective 3)

**msDate** LocalDate The milestone date.

#### 2.8.2 Backing Database Table

Each record in the backing table represents a single **StandardMilestoneRec** data object.

**Informix table name: math.std\_milestone**

**PostgreSQL table name: [term].std\_milestone**

**PostgreSQL fields:**

**pace\_track** char(1) The **paceTrack** field (Primary key).

**pace** smallint The **pace** field (Primary key).

**pace\_index** smallint The **paceIndex** field (Primary key).

**unit** smallint The **unit** field, like "2324" (Primary key).

**objective** smallint The **objective** field (Primary key).

**ms\_type** char(2) The **msType** field (Primary key).

**ms\_date** date The **msDate** field.

#### 2.8.3 Data Retention

Records are created in each term schema for that term's milestone dates, and are updated only in response to some event like a snow closure or system failure that warrants shifting deadlines globally.

#### 2.8.4 API

The API is exposed via the **StandardMilestoneLogic** utility class.

boolean insert(Cache cache, **StandardMilestoneRec** record);

boolean delete(Cache cache, **StandardMilestoneRec** record);

boolean updateDate(Cache cache, **StandardMilestoneRec** record, LocalDate newDate);

List<**StandardMilestoneRec**> queryAll(Cache cache);

List<**StandardMilestoneRec**> queryByPaceTrackPace(Cache cache, final String track, Integer pace);

List<**StandardMilestoneRec**> queryByPaceTrackPaceIndex(Cache cache, String track,Integer pace,  
 Integer paceIndex);  
 **StandardMilestoneRec** query(Cache cache, String track, Integer pace, Integer paceIndex, Integer unit,  
 Integer objective, String msType);

### 2.9 StudentStandardMilestoneRec [term]

Each object represents an override for a milestone date for a student in the current term.

#### 2.9.1 Member Fields

**stuId** String The student ID.

**paceTrack** String The pace track for which the milestone applies.

**pace** Integer The pace for which the milestone applies.

**paceIndex** Integer The pace index for which the milestone applies.

**unit** Integer The unit for which the milestone applies.

**objective** Integer The objective for which the milestone applies.

**msType** String The milestone type.

**msDate** LocalDate The override milestone date.

#### 2.9.2 Backing Database Table

Each record in the backing table represents a single **StandardMilestoneRec** data object.

**Informix table name: math.stu\_std\_milestone**

**PostgreSQL table name: [term].stu\_std\_milestone**

**PostgreSQL fields:**

**stu\_id** char(9) The **stuId** field (Primary key).

**pace\_track** char(1) The **paceTrack** field (Primary key).

**pace** smallint The **pace** field (Primary key).

**pace\_index** smallint The **paceIndex** field (Primary key).

**unit** smallint The **unit** field, like "2324" (Primary key).

**objective** smallint The **objective** field (Primary key).

**ms\_type** char(2) The **msType** field (Primary key).

**ms\_date** date The **msDate** field.

#### 2.9.3 Data Retention

Records are created whenever an override is granted to a student, in response to an SDC letter, letter from another student acvocacy group, or documented extenuating circumstance.

#### 2.9.4 API

The API is exposed via the **StudentStandardMilestoneLogic** utility class.

boolean insert(Cache cache, **StudentStandardMilestoneRec** record);

boolean delete(Cache cache, **StudentStandardMilestoneRec** record);

boolean updateDate(Cache cache, **StudentStandardMilestoneRec** record, LocalDate newDate);

List<**StudentStandardMilestoneRec**> queryAll(Cache cache);

List<**StudentStandardMilestoneRec**> queryByStudent(Cache cache, String stuId);

List<**StudentStandardMilestoneRec**> queryByStuPaceTrackPace(Cache cache, String stuId, String track,  
 Integer pace);

List<**StudentStandardMilestoneRec**> queryByStuPaceTrackPaceIndex(Cache cache, String stuId, String track,  
 Integer pace, Integer paceIndex);  
 **StudentStandardMilestoneRec** query(Cache cache, String stuId, String track, Integer pace,  
 Integer paceIndex, Integer unit, Integer objective, String msType);

### 2.1 Calendar [term]

Each object represents a significant date in the current term. Dates for future terms can be pre-populated and may be used in web sites to display (for example) when the center will re-open after a semester break. The three lists (openingTimeList, closingTimeList, and openWeekdaysList) must always hold the same number of entries.

#### 2.1.1 Member Fields

**calendarDate** LocalDate A date on the calendar

**calendarDateType** ECalendarDateType The date type.  
 DROP\_DATE - Last date when classes may be dropped.  
 START\_DATE\_1 - Start of the first interval of open days.  
 START\_DATE\_2 - Start of the second interval of open days.  
 END\_DATE\_1 - End of the first interval of open days.  
 END\_DATE\_2 - End of the second interval of open days.  
 START\_DATE\_1\_NEXT - Start of the first interval of open days next term.  
 END\_DATE\_1\_NEXT - End of the first interval of open days next term.  
 HOLIDAY - A holiday.  
 BOOKSTORE - The last date the bookstore will accept returns of e-texts.  
 TUT\_START - The first date tutorial sites will be open.  
 TUT\_END - The last date tutorial sites will be open.  
 TUT\_INFO - When tutorial information becomes available.  
 WALKIN\_PLACEMENT - Walk-in placement exam session.  
 PLACEMENT\_MAINT\_START1 - Start of 1st placement maint. window.  
 PLACEMENT\_MAINT\_END1 - End of 1st placement maint. window.  
 PLACEMENT\_MAINT\_START2 - Start of 2nd placement maint. window.  
 PLACEMENT\_MAINT\_END2 - End of 2nd placement maint. window.  
 PLACEMENT\_MAINT\_START3 - Start of 3rd placement maint. window.  
 PLACEMENT\_MAINT\_END3 - End of 3rd placement maint. window.  
 SM\_LAST\_ADD - Last date to add Summer courses (only present in SM).

**openingTimeList** List<LocalTime> The list of opening times. Non-empty for the START\_DATE\_1,  
 START\_DATE\_2, and WALKIN\_PLACEMENT calendar date types.

**closingTimeList** List<LocalTime> The list of closing times, non-empty only if **openingTimeList** is.

**openWeekdaysList** List<Integer> The list of open weekdays, non-empty only if **openingTimeList** is. A bitwise  
 OR of bits representing weekdays:  
 0x01=Sunday, 0x02 = Monday, 0x04 = Tuesday, 0x08 = Wednesday,  
 0x10 = Thursday, 0x20 = Friday, 0x40 = Saturday

#### 2.1.2 Backing Database Table

Each record in the backing table represents a single **Calendar** data object.

**Current table name: math.campus\_calendar**

**Proposed name:**  **[term].campus\_calendar**

**Fields:**

**campus\_dt** date, not null  
 The **calendarDate** field.

**dt\_desc** char(20), not null  
 The **calendarDateType** field. Contains one of the following text strings:  
 drop\_dt  
 start\_dt1  
 start\_dt2  
 end\_dt1  
 end\_dt2  
 start\_dt1x  
 end\_dt1x  
 holiday  
 bookstore  
 tut\_start  
 tut\_end  
 tut\_info  
 walk\_in  
 plc\_maint\_start1  
 plc\_maint\_end1  
 plc\_maint\_start2  
 plc\_maint\_end2  
 plc\_maint\_start3  
 plc\_maint\_end3  
 sm\_last\_add  
 Proposed name: **dt\_type**.

**open\_time1** char(10)  
 The first entry in the **openingTimeList** field.  
 Proposed data type: time rather than string like "9:00 am"

**open\_time2** char(10)  
 The second entry in the **openingTimeList** field.  
 Proposed data type: time rather than string like "6:30 pm"

**close\_time1** char(10)  
 The first entry in the **closingimeList** field.  
 Proposed data type: time rather than string like "4:00 pm"

**close\_time2** char(10)  
 The second entry in the **closingimeList** field.  
 Proposed data type: time rather than string like "9:00 pm"

**weekdays1**  char(20)  
 The first entry in the **openWeekdaysList** field.  
 Proposed data type: int with bit flags rather than string like "Monday - Friday"

**weekdays2** char(20)  
 The second entry in the **openWeekdaysList** field.  
 Proposed data type: int with bit flags rather than string like "Monday - Thursday"

#### 2.1.3 Term Rollover

The current table stores all records for the current term, and some records from at least two future terms. It is the responsibility of query code to filter data by term if needed.

The proposed tables in term databases would be populated as soon as the term data was available (but not less than two terms ahead) and would simply remain in place as the term databases became archival databases.

#### 2.1.4 API

The API is exposed via the "CalendarCache" utility class.

List<Calendar> queryAll();

List<Calendar> queryByType(ECalendarDateType theType);

LocalDate getFirstClassDay();

LocalDate getLastClassDay();

### 2.2 CatalogCourse [common]

Each object represents a course in the University catalog. This data (and the prerequisites associated with each course) is used by the "Math Plan" web site to determine, for each major, the mathematics courses that are required or recommended.

#### 2.2.1 Member Fields

**academicYear** String The academic year, such as "1920".

**courseNumber** String The course number, such as "MATH 117", truncated to 12 characters if  
 necessary. May not contain commas.

**title** String The catalog title.

**minCredits** Integer The minimum number (or fixed number) of credits.

**maxCredits** Integer The maximum number (or fixed number) of credits.

**prereqString** String The prerequisite string, as shown in the catalog. This is a freeform  
 string from which one can try to parse prerequisites.

**description** String The catalog description, null if none.

#### 2.2.2 Backing Database Table

Each record in the backing table represents a single **CatalogCourse** data object.

**Current table name: (None)**

**Proposed name:**  **[common].catalog\_course**

**Fields:**

**academic\_yr**  char(4), not null  
 The **academicYear** field.

**course\_nbr**  char(12), not null  
 The **courseNumber** field.

**title**  varchar(80), not null  
 The **title** field.

**min\_credits**  smallint, not null  
 The **minCredits** field.

**max\_credits**  smallint, not null  
 The **maxCredits** field.

**prereq\_str** varchar(250), not null  
 The **prereqString** field.

**description** text  
 The **description** field.

#### 2.2.3 Term Rollover

The proposed tables in the common databases would remain in place indefinitely, being updated from time to time as the catalog is re-scanned. Term tables could have a “snapshot” of this table as an archive.

#### 2.2.4 API

The API is exposed via the "CatalogCourseCache" utility class.

List<CatalogCourse> queryAll();

### 2.3 CatalogCourseGroup [common]

Each object represents a group of courses within a prerequisite definition, where students choose one course or a specified number of credits from courses in the group. A course catalog course may be considered a course group with only one course. Course group codes must not match a course number, since prerequisite tracks may refer to courses or to course groups.

#### 2.3.1 Member Fields

**groupCode** String The group code.

**numCredits** Integer The number of credits needed, null if student must choose one course.

**lastEasyCourse** String The highest course number in the **courseNumbers** list that is considered "easy". Courses in the course numbers list should be sorted (crudely) from easiest to hardest, and this course number marks the last of the "easy" category. This is used, for example, when generating a curriculum plan - students needing to take N credits or 1 course from a list might be assumed to be more likely to take easy courses.

**courseNumbers** List<String> A list of course numbers in the group, like "MATH 117", sorted from easiest  
to hardest. Each course in this list should exist as a **CatalogCourse** object.

#### 2.3.2 Backing Database Table

Each record in the backing table represents a single **CatalogCourseGroup** data object.

**Current table name: (none)**

**Proposed name:**  **[common].catalog\_course\_group**

**Fields:**

**course\_grp**  varchar(10), not null  
 The **groupCode** field.

**nbr\_credits**  smallint  
 The **numCredits** field.

**last\_easy\_course** char(12)  
 The **lastEasyCourse** field.

**course\_nbrs** varchar(250)  
 The **courseNumbers** field, stored as a comma-separated list of course numbers.

#### 2.3.3 Term Rollover

The proposed table in the common databases would remain in place indefinitely, being updated from time to time as the catalog is re-scanned. Term tables could have a “snapshot” of this table as an archive.

#### 2.3.4 API

The API is exposed via the "CatalogCourseGroupCache" utility class.

List<CatalogCourseGroup> queryAll();

CatalogCourseGroup query(String groupCode);

### 2.4 CatalogCoursePrereqTrack [common]

Each object is one possible way to satisfy the prerequisites for a course. The combination of academic year and course number refer to a **CatalogCourse** object (a foreign key relationship).

#### 2.4.1 Member Fields

**academicYear** String The academic year, such as "1920".

**courseNumber** String The course number, such as "MATH 117", truncated to 12 characters if  
 necessary. May not contain commas.

**prereqCourses** List<String> A list of course numbers or course group IDs, all of which must be  
 completed to satisfy prereq.

**percentage** Double The estimated percentage of students using this track to satisfy prereqs.  
 Null if not known.

#### 2.4.2 Backing Database Table

Each record in the backing table represents a single **CatalogCoursePrereqTrack** data object.

**Current table name: (None)**

**Proposed name:**  **[common].catalog\_course\_preq**

**Fields:**

**academic\_yr**  char(4), not null  
 The **academicYear** field.

**course\_nbr**  char(12), not null  
 The **courseNumber** field.

**prereqs** varchar(250), not null  
 The **prereqCourses** field, stored as a comma-separated list of course numbers.

**percentage** float4  
 The **percentage** field.

#### 2.4.3 Term Rollover

The proposed tables in the common databases would remain in place indefinitely, being updated from time to time as the catalog is re-scanned. Term tables could have a “snapshot” of this table as an archive.

#### 2.4.4 API

The API is exposed via the "CatalogCoursePrereqTrackCache" utility class.

List<CatalogCoursePrereqTrack> queryAll();

List<CatalogCoursePrereqTrack> queryByCourse(String courseNbr);

### 2.5 ChallengeFee [common]

Each object is a record of a fee assessed to a student for a challenge exam. At present there should be no more than one object for a particular combination of student ID and course ID, since students have only one attempt on each course's challenge exm. However, to allow for changes in that policy in the future, we include exam date in the primary key. This would limit students to one attempt per day.

#### 2.5.1 Member Fields

**studentId** String The ID of the student to whom fee was assessed.

**courseId** String The ID of the course that was challenged, of the form "M 117".

**examDate** LocalDate The date when the exam was taken.

**billDate** LocalDate The date when the fee was billed to the student account.

#### 2.5.2 Backing Database Table

Each record in the backing table represents a single **ChallengeFee** data object.

**Current table name: math.challenge\_fee**

**Proposed name:**  **[common].challenge\_fee**

**Fields:**

**stu\_id**  char(9), not null  
 The **studentId** field.

**course\_id**  char(6), not null  
 The **courseId** field.

**exam\_dt** date, not null  
 The **examDate** field.

**bill\_dt** date  
 The **bilLDate** field.

#### 2.5.3 Term Rollover

Data must be stored long enough to ensure a student is not charged twice. At least 10 years, 20 years preferred.

#### 2.5.4 API

The API is exposed via the "ChallengeFeeCache" utility class.

ChallengeFee query(String studentId, String courseId);

boolean insert(ChallengeFee obj);

### 2.6 Cohort [term]

Each object represents a cohort of students who are all registered for the same combination of courses, and who are completing those courses in the same order (so they will all reach the same course content at the same time). If many students have the same course combination, multiple cohorts may be defined to achieve a target maximum cohort size. Cohorts may be used to simulate small-class dynamics like class fora and discussions, or social-media style interactions between students or with the instructional team.

#### 2.6.1 Member Fields

**cohortId** String The cohort ID, which is a string of the form “78456.1”, limited to 8  
 characters in length. The digits before the radix are the trailing digits of  
 course IDs, and the integer after the radix is an index of cohort within a  
 course combination (these begin at 1 and increment for each cohort  
 created). This format allows up to 999,999 cohorts of each 1-course pace,  
 99,999 of each 2-course, 9,999 of each 3-course, 999 of each 4-course, and  
 99 of the 5-course pace.

**size** Smallint The number of students currently in the cohort.

**instructor** String The instructor’s name, in the form “Benoit, Steve”, null if no instructor has  
 been assigned.

#### 2.6.2 Backing Database Table

Each record in the backing table represents a single **Cohort** data object.

**Current table name: math.cohort**

**Proposed name:**  **[term].cohort**

**Fields:**

**cohort\_id**  char(8), not null  
 The **cohortId** field.

**size**  smallint, not null  
 The **size** field.

**instructor** varchar(80)  
 The **instructor** field.

#### 2.6.3 Term Rollover

Cohorts are assigned fresh each term and may be updated during the term if a student’s registrations change. They are retained in term databases as an archival record, but not carried forward from term to term. Once a cohort is created, it is not deleted during its term, even if all students move out of the cohort by changing their registrations. This implies size may be zero.

#### 2.6.4 API

The API is exposed via the "CohortCache" utility class.

Cohort query(String cohortId);

boolean insert(Cohort obj);

boolean updateCSize(String cohortId, int newSize);

### 2.7 Course [common]

Each object represents a course that is offered through the system. These can be real courses, for which students register and that appear in the University catalog, or can be "bogus" courses, used as place-holders for offerings like the placement tool, non-credit tutorials, etc.

#### 2.7.1 Member Fields

**courseId** String The course ID, such as "M 117".

**title** String The course title, such as "College Algebra I", null if not relevant.

**courseLabel** String The course label, which is a course number like "MATH 117" for courses in  
 the catalog, or a text label like "Math Placement" or "ELM Tutorial" for a  
 "bogus" course.

**inlinePrefix** String? A prefix to use when the coursed label is used in body text. For example,   
 for a course with label "MATH 117, this would be null, and for a course  
 with label "ELM Tutorial", this could be "The". This would generate   
 displays like "when completing MATH 117, word hard." and "when   
 completing the ELM tutorial, work hard.". This should be stored with   
 leading uppercase letter - if used in body text in a context where lowercase   
 is apropriate, it can be converted to lowercase, and when used as a   
 heading where all caps is apropriate, it can be converted to caps.

**catalogUrl** String? The URL of the course page in the catalog.

**numUnits** Integer The number of units in the course. Currently 1 for courses that don't have  
 actual units; may be null for these courses in the future.

**numCredits** Integer The number of credits (null if variable).

**isTutorial** Boolean A flag indicating the course is a tutorial.

**isCalculatorOk** Boolean A flag indicating it is OK to issue the student a calculator during exams in  
 the testing center.

**requirePlacement** Boolean A flag indicating course requires students to have completed placement.

**requireEText** Boolean A flag indicating the course requires an e-text.

#### 2.7.2 Backing Database Table

Each record in the backing table represents a single **Course** data object.

**Current table name: math.course**

**Proposed name:**  **[common].course**

**Fields:**

**course**  char(6), not null  
 The **courseId** field.

**nbr\_units** smallint, not null  
 The **numUnits** field.  
 Proposed that this be nullable for courses like "Math Placement" where it is not relevant

**course\_name** varchar(80)  
 The **title** field.

**nbr\_credits** smallint, not null  
 The **numCredits** field.  
 Proposed that this be nullable, to allow for variable-credit courses.

**calc\_ok** char(1)  
 The **isCalculatorOk** field. True stored as 'Y', false stored as 'N', currently this field allows nulls  
 which are interpreted as 'N'.  
 Proposed that this not be nullable, either 'Y' or 'N'.

**course\_label** char(40)  
 The **courseLabel** field.  
 Proposed data type: varchar(40)

**inline\_prefix** char(20)  
 The **inlinePrefix** field.  
 Proposed data type: varchar(20)

**is\_tutorial** char(1), not null  
 The **isTutorial** field. True stored as 'Y', false stored as 'N'.

**require\_etext** char(1), not null  
 The **requireEtext** field. True stored as 'Y', false stored as 'N'.

**require\_placement** char(1), not null  
 Proposed to store the **requirePlacement** field, which is now hard-coded. True stored as 'Y', false  
 stored as 'N'.

**catalog\_url**  varchar(250)  
 Proposed to store for the the **catalogUrl** field.

#### 2.7.3 Term Rollover

Records remain in the table indefinitely, updated manually as courses are changed.

#### 2.7.4 API

The API is exposed via the "CourseCache" utility class.

Course getCourse(String courseId);

### 2.8 CourseSection [term]

Each object represents a single section of a course offered in the term. Note that this includes "bogus" courses like those that represent math placement, tutorials, etc.

#### 2.8.1 Member Fields

**term** TermKey The term key.

**courseId** String The course ID.

**sectionNum** String The section number, like "001".

**sectionCode** String? The internal section code (CRN) used to submit grades.

**ariesStartDate** LocalDate? The section start date as listed in University sustems (this is what should be  
 shown to students as the start date).

**ariesEndDate** LocalDate? The section end date as listed in University sustems (this is what should be  
 shown to students as the end date).

**startDate** LocalDate? The section start date at which time exams can be taken. This allows the  
 system to be opened for testing before a published "start date" from  
 University systems.

**examDeleteDate** LocalDate? The date when exams for the section may be deleted.

**lastStudentAddDate** LocalDate? The last date when students may add the course without special override.

**instructionType** EInstructionType? The instruction type:  
 AP - Advance placement  
 OT - Credit by challenge exam  
 RI - Resident instruction  
 CE - Continuing education  
 CR - Continuous registration  
 CT - Unknown code

**examStructure** EExamStructure? The exam structure:  
 UNIT\_FINAL - Unit and final exams  
 UNIT\_ONLY - Only unit exams  
 FINAL\_ONLY - Only a final exam

**ruleSetId** String? The ID of the rule set with the section policies and rules.

**isTopmatterInPractice** Boolean? True if section top matter should be shown in practice mode.

**gradingSystemId** String? The grading system.

**minScoreForA** Integer? The minimum score to earn an A grade.

**minScoreForB** Integer? The minimum score to earn an B grade.

**minScoreForC** Integer? The minimum score to earn an C grade.

**minScoreForD** Integer? The minimum score to earn an D grade.

**countForMaxOpen** Boolean? Flag indicating the course is counted toward maximum open courses.

**isCourseLabelShown** Boolean? Flag indicating the course label appears in the course menu.

**isScoreShown** Boolean? Flag indicating the course score should be displayed.

**isGradingScaleShown** Boolean? Flag indicating the course grading scale should be displayed.

**isDistance** Boolean Flag indicating this is a distance section.

**isOnline** Boolean Flag indicating this in an online section.

**isBogus** Boolean Flag indicating this in a bogus section.

**proctoringOptions** List<EProctoringOption>? The list of proctoring options to display.  
 DEPT\_TEST\_CENTER  
 UNIV\_TEST\_CENTER  
 ASSIST\_TEST\_CENTER  
 DIST\_TEST\_CENTER  
 HUMAN  
 PROCTOR\_U\_STUDENT  
 PROCTOR\_U\_UNIV  
 DEPT\_ONLINE

**topmatter** String? HTML top matter to include in course outline page.

#### 2.8.2 Backing Database Table

Each record in the backing table represents a single **Course** data object.

**Current table name: math.csection**

**Proposed name:**  **[term].csection**

**Fields:**

**course**  char(6), not null  
 The **courseId** field.

**sect**  char(4), not null  
 The **sectionNum** field.

**term**  char(2), not null  
 The term name component of the **term** field ('FA', 'SP', or 'SM')

**term\_yr**  smallint, not null  
 The term year component of the **term** field (the short year, add 2000 to all values less than 70,  
 add 1900 to all values 70 or greater).

**section\_id** char(6)  
 The **sectionCode** field.

**aries\_start\_dt** date  
 The **ariesStartDate** field.

**aries\_end\_dt** date  
 The **ariesEndDate** field.

**start\_dt**  date  
 The **startDate** field.

**exam\_delete\_dt** date  
 The **examDeleteDate** field.

**instrn\_type** char(2)  
 The **instructionType** field. One of the following constants: 'AP', 'OT', 'RI', 'CE', 'CR', 'CT'.

**instructor** char(30)  
 Not currently mapped to a data field - the instructor assigned to the section.

**campus** char(2), not null  
 Not currently mapped to a data field - the campus where the section is to be taught.

**pacing\_structure** char(1)  
 The **ruleSetId** field.

**mtg\_days** char(5)  
 Not currently mapped to a data field - the days of the week this section is taught.

**classroom\_id** char(14)  
 Not currently mapped to a data field - the ID of the classroom where the section is taught.

**last\_stcrs\_creat\_dt** date  
 The **lastStudentAddDate** field.

**grading\_std** char(3)  
 The **gradingSystemId** field.

**a\_min\_score** smallint  
 The **minScoreForA** field.

**b\_min\_score** smallint  
 The **minScoreForB** field.

**c\_min\_score** smallint  
 The **minScoreForC** field.

**d\_min\_score** smallint  
 The **minScoreForD** field.

**survey\_id** char(5)  
 Not currently mapped to a data field - the ID if a course survey.

**couse\_label\_shown** char(1)  
 The **isCourseLabelShown**field. True stored as 'Y', false as null or 'N'.

**display\_score** char(1)  
 The **isScoreShown** field. True stored as 'Y', false stored as null or 'N'.

**display\_grade\_scale** char(1)  
 The **isGradingScaleShown** field. True stored as 'Y', false stored as null or 'N'.

**count\_in\_max\_courses** char(1)  
 The **countForMaxOpen** field. True stored as 'Y', false stored as null or 'N'.

**online** char(1), not null  
 The **countForMaxOpen** field. True stored as 'Y', false stored as 'N'.

**bogus** char(1), not null  
 The **countForMaxOpen** field. True stored as 'Y', false stored as 'N'.

#### 2.8.3 Term Rollover

Records are currently created for each new term, and are retained for at least 16 years. When data is stored in term database, only that term's course sections will be stored, with historic data available in the prior term databases.

#### 2.8.4 API

The API is exposed via the "CourseSectionCache" utility class.

List<CourseSection> getAllSections(TermKey term);

CourseSection getCourseSection(String courseId, String section, TermKey term);

### 2.10 CourseSectionUnit [term]

Each object represents the configuration of a single unit within a single section of a course offered in the term.

**term** TermKey The term data.

**courseId** String The course ID.

**sectionNum** String The section number, like "001".

**unit** Integer The unit number.

**skillsReviewMaxScore** Integer? The maximum possible score on the skills review exam.

**skillsReviewMastery** Integer? The mastery score on the skills review exam.

**unitReviewMaxScore** Integer? The maximum possible score on the unit review exam.

**unitReviewMastery** Integer? The mastery score on the unit review exam.

**unitExamMaxScore** Integer? The maximum possible score on the unit exam.

**unitExamMastery** Integer? The mastery score on the unit exam.

**finalExamMaxScore** Integer? The maximum possible score on the final exam.

**finalExamMastery** Integer? The mastery score on the final exam.

**homeworkMaxScore** Integer The maximum possible score on homework sets.

**homeworkMastery** Integer The mastery score on homework sets.

**homeworkMoveOn** Integer The minimum score needed to move beyond a homework set.

**unitTriesAllowed** Integer The maximum attempts allowed on unit exams.

**unitTriesPerReview** Integer The unit exam attempts allowed per passing review exam.

**showTestingWindow** Boolean? Flag indicating testing window is displayed.

**firstTestDate** LocalDate The first date a proctored exam may be taken.

**lastTestDate** LocalDate The last date a proctored exam may be taken.

**testPeriodStartTime** LocalTime The first time of day a proctored exam may be taken in a testing center.

**testPeriodEndTime** LocalTime The last time of day a proctored exam may be taken in a testing center.

**hasUnproctoredExam** Boolean? Flag indicating "proctored" exam does not need to be proctored.

**reviewPointsOntime** Integer? The number of points earned by mastering the review exam on time.

**topmatter** String? HTML top matter to include in course outline page.

**lessonCoureId** String? The ID of a course to be used as the lesson.

### 2.11 CourseUnit [term]

Each object represents the section-independent configuration of a single unit within a single course.

**term** TermKey The term data.

**courseId** String The course ID.

**unit** Integer The unit number.

**title** String The unit title.

**unitType** EUnitType The unit type:

INSTRUCTION

SKILLS\_REVIEW

FINAL

EXAM

**possibleScore** Integer The highest possible score on the unit exam.

### 2.12 CourseUnitObjective [term]

Each object represents one objective (with associated homework set) in a course unit.

**term** TermKey The term data.

**courseId** String The course ID.

**unit** Integer The unit number.

**objective** Integer The objective number.

**lessonId** String The lesson ID.

**lessonNumber** String The lesson number.

### 2.14 DuplicateRegistration [term]

Each object represents ...

### 2.15 EText [common]

Each object represents one e-text that students can purchase to access course materials.

**eTextId** String The e-text ID.

**retention** EETextRetention The retention policy:

YES - access retained indefinitely regardless of course completion

COMPLETED - access retained for completed courses

NO - Access not retained into future terms

**purchaseUrl** String The URL at which to purchase the e-text.

**refundPeriod** Integer The refund period, in days.

**allowKeyEntry** Boolean Flag that allows users to enter keys.

**active** Boolean Flag that indicate e-text is active.

**description** String Description of the e-text.

### 2.16 ETextCourse [common]

Each object represents one course that owners of an e-text can access.

**eTextId** String The e-text ID.

**courseId** String The course ID.

### 2.17 ETextKey [common]

Each object represents one access code/key associated with a purchase of an e-text. Students can type in these keys to active their purchased e-text.

**eTextKey** String The key.

**eTextId** String The e-text ID.

**whenActive** LocalDateTime The date/time the key was activated (and StudentEText record added)

### 2.18 Exam [common]

Each object represents an exam that can be used in a course or tutorial.

NOTE: Final grading program assumes all review exams will have an exam ID that ends in "RE", and that all exams with ID that ends in "RE" are review exams (where completion of the exam on time can give points).

**examId** String The exam ID.

**examType** EExamType The exam type.

SKILLS\_REV

UNIT

UNIT\_REV

FINAL

LICENSING

PLACEMENT

CHALLENGE

**courseId** String The course ID for which the exam is intended.

**unit** Integer The course unit for which the exam is intended.

**treeRef** String The tree reference.

**title** String The exam title.

**buttonLabel**  String The exam button label.

**whenActive** LocalDateTime The date/time the exam was made active.

**whenPulled** LocalDateTime The date/time the exam was pulled from service.

### 2.19 GradingStandard [common]

Each object represents ...

### 2.20 GuestStudent [common]

Each object represents ...

### 2.21 HighSchool [common]

Each object represents a high school. Used in annual reports to Colorado high schools regarding placement rates.

**hsCode** String The high school code.

**hsName** String The high school name.

**address** String The street address.

**city** String The city.

**state** String The state.

**zipCode** String The zip code

### 2.22 HoldType [common]

Each object represents a type of hold that can be applied to a student account.

**holdId** String The hold ID.

**holdSeverity** String The severity ('F' if fatal, 'N' if nonfatal**)**

**holdCategory** String The hold category.

**staffCanAdd** Boolean True if staff can apply the hold.

**staffCanDelete** Boolean True if staff can delete the hold (unless this is a resource or rental  
 hold type and has been applied the number of times defined for  
 that resource).

**description** String The description of the hold, to display to staff.

**messageStudent** String The message to display to the student.

Hold IDs include:

01 (F, REGIS, N, N) unauthorized grading option selected by student - have student talk w/ director

02 (N, REGIS, N, Y) warn student they are registered for more than recommended # of courses in SM

03 (F, REGIS, N, N) registered for both course and placement section - refer to a director!

04 (F, REGIS, Y, N) can never reregister for course -- see discipline table for details

05 (F, ADMIN, Y, N) deferred F issued to student -- see discipline table for details

06 (F, ADMIN, Y, N) discretionary administrative hold -- have student speak with a director

07 (F, USERS, N, Y) profile info specifies different registrtion format than is in STUDENT record

08 (N, PACING, Y, Y) student is behind course pacing schedule -- WARN STUDENT ACCORDINGLY

09 (N, PACING, Y, Y) student is behind unit pacing schedule --- WARN STUDENT ACCORDINGLY

10 (F, USERS, N, Y) student has mixed instr and stu managed courses, registering thru both RI and CE

11 (F, ADMIN, Y, N) answer sheet not turned in

12 (F, MISCS, N, Y) conditionally issued exam not verified

13 (F, BATCHFIX, N, N) licensed, but not registered in section for which batch exam was administered

14 (F, REGIS, Y, Y) duplicate RI & CE registration for same course -- CHECK ARIES!

15 (F, USERS, N, Y) User's survey says student is taking both stu managed and instructor led

16 (F, REGIS, Y, Y) duplicate 001 & 401 registration for same course - CHECK ARIES!

18 (F, MPE, Y, Y) submitted an MPE attempt that cannot be counted

19 (F, BATCHFIX, N, N) not licensed and not registered in sect for which batch exam was administered

20 (F, BATCHFIX, N, N) course prereq not satisfied for an exam taken during a batch administration

21 (F, USERS, N, Y) tried to create stcourse record after deadline date based on User's survey

22 (F, USERS, N, Y) tried to create stu managed stcourse rowafter add deadline date so not registerd

23 (F, REGIS, N, Y) officially registered for two different PACe formats -- needs to change registration

24 (F, BATCHFIX, N, N) tried to take batch exam for course/sectin which open\_status = 'N'

25 (F, REGIS, N, Y) student is working on an INC but has re-registered for course in error

26 (F, REGIS, N, Y) registered for placement credit that hasbeen declined

27 (F, REGIS, N, Y) has registered for placement credit but is not eligible

28 (N, USERS, N, Y) \*\*\* 4 WEEK SUMMER STUDENT \*\*\* GIVE STUDENT SPECIAL GRADING HANDOUT

29 (F, REGIS, N, Y) \*\*\* M120B student who is also registeredfor another minicourse -- PROBLEM!!! \*\*\*

30 (F, LOCKOUT, Y, N) student has not met weekly progress requirement and may no longer test

31 (N, SPECIAL, N, N) have student talk with a director

32 (N, ADMIN, Y, Y) student has lost their ID -- ask director to check collection box

33 (N, ADMIN, Y, Y) student may have lost personal items -- check lost/found box & ask director

34 (F, SPECIAL, N, N) have student talk with Anita or Steve

36 (N, RENTAL, N, Y) REMIND STUDENT: their RENTAL CALC needs to be returned by the last day of finals

41 (F, RESOURCE, Y, Y) student has an OVERDUE IN-HOUSE VIDEO --ask to return immediately!

42 (F, RESOURCE, Y, Y) student has an OVERDUE IN-HOUSE TEXT -- ask to return it or check it out again

43 (F, RESOURCE, Y, Y) student has a LATE OVERNIGHT VIDEO -- ask to return by the end of the day!

44 (F, RESOURCE, Y, Y) student has a LATE OVERNIGHT TEXTBOOK - ask to return by the end of the day!

45 (F, RESOURCE, Y, Y) student has an OVERDUE IN-HOUSE CALC -- ask to return immediately!

46 (F, RENTAL, N, N) student has an OVERDUE RENTAL CALC -- must be returned immediately!

47 (F, RENTAL, N, N) student has an OVERDUE RENTAL CALC MANUAL -- must be returned immediately!

48 (N, RENTAL, Y, Y) Student has rented a calc --> SEND TO OFFICE for copy of RENTAL AGREEMENT

49 (F, ADMN, Y, Y) Student needs to get a new CSU ID card because we cannot identify properly

50 (F, ADMIN, Y, Y) FINAL WARNING to student: return TI-83 calc or be billed $100 replacement fee

51 (F, RESOURCE, Y, Y) student has OVERDUE HEADPHONES -- ask to return immediately!

52 (F, RESOURCE, Y, Y) student has an OVERDUE PADLOCK -- ask to return it immediately!

53 (F, RESOURCE, Y, Y) student has an OVERDUE IPAD -- ask to return it immediately!

54 (F, RESOURCE, Y, Y) student has an OVERDUE LAPTOP -- ask to return it immediately!

### 2.24 Lesson [common]

Each object represents a lesson that can be used within a course.

**lessonId** String The lesson ID.

**description** String The description.

### 2.25 LessonComponent [common]

Each object represents a component that can be used within a lesson.

**lessonId** String The lesson ID.

**sequeneNumber** Integer The sequence number (to order within the lesson).

**type** String The component type.

**xmlData** String The XML data.

### 2.26 LiveAdvisee [N/A]

Each object represents a student assigned as an advisee to an adviser.

**studentId** String The student ID.

**studentPidm** String The system internal ID of the student.

**studentFirstName** String The student's first name.

**studentLastName** String The student's last name.

**adviserId** String The adviser's ID.

**adviserPidm** String The system internal ID of the adviser.

**adviserFirstName** String The adviser's first name.

**adviserLastName** String The adviser's last name.

### 2.27 LiveCSUCredit [N/A]

Each object represents a course in which a student earned credit at CSU.

**studentId** String The student ID.

**term** TermKey The term when credit was earned.

**course** String The course number.

**courseId** String The course ID.

**credits** Double The number of credits earned.

**grade** String The earned grade.

**gradeGpa** String The grade, converted to a GPA value if passing.

### 2.28 LiveReg [N/A]

Each object represents a record of a student's registration in a course during the current semester.

**term** TermKey The term.

**anticGradTerm** TermKey The term the student anticipates graduating.

**studentId** String The student ID.

**internalId** Integer The internal ID of the student.

**lastName** String The student's last name.

**firstName** String The student's first name.

**classLevel** String The class level.

**college** String The student's home college.

**department** String The student's home department.

**major1** String The student's declared/primary major.

**numTransferCredits** String The number of transfer credits.

**highSchoolCode** String The student's high school code.

**highSchoolGpa** String The student's GPA from high school.

**highSchoolClassRank** Integer The student's class rank in high school.

**highSchoolClassSize** Integer The student's class size in high school.

**actScore** Integer The student's ACT math score.

**satScore** Integer The student's SAT math score.

**satrScore** Integer The student's revised SAT math score.

**apScore** Integer The student's AP math score.

**residency** String The student's residency.

**birthDate** LocalDate The student's birth date.

**gender** String The student's gender.

**email** String The student's email address.

**adviserEmail** String The student's adviser's email address.

**campus** String The campus to which the student is assigned.

**admitType** String The admission type.

**courseId** String The ID of the course in which the student is registered.

**sectionNum** String The section number of the course in which the student is registered.

**gradingOption** String The selected grading option.

**registrationStatus** String The student's registration status.

**instructionType** EInstructionType The instruction type:

AP - Advance placement

OT - Credit by challenge exam

RI - Resident instruction

CE - Continuing education

CR - Continuous registration

CT - Unknown code

### 2.29 LiveStudent [N/A]

Each object represents a student.

**studentId** String The student ID.

**internalId** Integer The internal ID of the student.

**lastName** String The student's last name.

**firstName** String The student's first name.

**prefName** String The student's preferred first name.

**middleInitial** String The student's middle initial.

**collegeCode** String The student's home college.

**departmentCode** String The student's home department.

**programCode** String The student's program of study.

**majorCode** String The student's primary major.

**minorCode** String The student's primary minor.

**highSchoolCode** String The student's high school code.

**highSchoolGpa** String The student's GPA from high school.

**highSchoolClassRank** Integer The student's class rank in high school.

**highSchoolClassSize** Integer The student's class size in high school.

**actScore** Integer The student's ACT math score.

**satScore** Integer The student's SAT math score.

**satrScore** Integer The student's revised SAT math score.

**apScore** Integer The student's AP math score.

**residency** String The student's residency.

**admitTerm** TermKey The term of the student's admission.

**admitType** String The admission type.

**estGradDate** LocalDate The estimated graduation date.

**birthDate** LocalDate The student's birth date.

**gender** String The student's gender.

**email** String The student's email address.

**adviserEmail** String The student's adviser's email address.

**state** String The student's state.

**campus** String The campus to which the student is assigned.

### 2.30 LiveTransferCredit [N/A]

Each object represents a record of credit from a transfer course for a student.

**studentId** String The student ID.

**term** TermKey The term transfer credit was earned.

**transferred** String The course number at the origin institution.

**courseId** String The course ID that the course transferred as.

**credits** Double The number of credits transferred.

**grade** String The transferred grade.

**gradeGpa** Float The grade, converted to a GPA value if passing.

### 2.31 LocalLogin [common]

Each object represents a login that can be used in lieu of an eID for sites that support such logins.

**username** String The login username.

**studentId** String The student ID.

**defaultRoleId** String The role under which the student logs in by default.

**passwordSalt** String The password salt.

**passwordHash** String The password hash.

**whenCreated** LocalDateTime The date/time the login record was created.

**whenLastLogin** LocalDateTime The date/time the user last logged in.  
**email** String The email to use for password resets.

**passwordFailsAllowed** Integer The number of consecutive password failures allowed.

**passwordFailsSoFar** Integer The number of consecutive password failures so far.

**disabled** Boolean True if login is disabled.

**whyDisabled** String The reason the login is disabled.

### 2.32 Message [term]

Each object represents a stock message that can be sent to students. Messages can be tailored each term.

**term** TermKey The term.

**touchPoint** ETouchPoint The touch point at which the message is sent:

TP\_1R1 - Course 1, prior to Review Exam 1 deadline

TP\_1R3 - Course 1, prior to Review Exam 3 deadline

TP\_1FE - Course 1, prior to Final Exam deadline

TP\_1F1 - Course 1, after Final Exam deadline, before F1 deadline

TP\_1BL - Course 1, after F1 deadline

TP\_2R1 - Course 2, prior to Review Exam 1 deadline

TP\_2R3 - Course 2, prior to Review Exam 3 deadline

TP\_2FE - Course 2, prior to Final Exam deadline

TP\_2F1 - Course 2, after Final Exam deadline, before F1 deadline

TP\_2BL - Course 2, after F1 deadline

TP\_3R3 - Course 3, prior to Review Exam 3 deadline

TP\_3FE - Course 3, prior to Final Exam deadline

TP\_3F1 - Course 3, after Final Exam deadline, before F1 deadline

TP\_3BL - Course 3, after F1 deadline

TP\_4R3 - Course 4, prior to Review Exam 3 deadline

TP\_4FE - Course 4, prior to Final Exam deadline

TP\_4F1 - Course 4, after Final Exam deadline, before F1 deadline

TP\_4BL - Course 4, after F1 deadline

TP\_5R3 - Course 5, prior to Review Exam 3 deadline

TP\_5FE - Course 5, prior to Final Exam deadline

TP\_5F1 - Course 5, after Final Exam deadline, before F1 deadline

**messageCode** String The message code.

**subject** String The subject line.

**template** String The message body template.

### 2.33 MessageLookup [common]

Each object represents...

### 2.34 Milestone [term]

Each object represents a milestone date within a pace and pace track.

**term** TermKey The term.

**pace** Integer The pace.

**paceTrack** String The pace track.

**milestoneNumber** Integer The milestone number (3-digit: pace/course/unit)

**milestoneType** EMilestoneType The milestone type:

SKILLS\_REVIEW

UNIT\_REVIEW\_EXAM

UNIT\_EXAM

FINAL\_EXAM

FINAL\_LASTTRY

**milestoneDate** LocalDate The milestone date.

**attemptsAllowed** Integer The number of attempts allowed; null if unlimited.

**courseIndex** Integer The course index (1 for first course, etc.).

**unit** Integer The unit.

### 2.35 NewStudent [term]

Each object represents...

### 2.36 PaceTrackRule [term]

Each object represents a rule that can assign a student to a pace track. These rules can change term to term.

**term** TermKey The term.

**paceTrack** String The pace track.

**paceTrackRule** String The rule expression.

### 2.37 Parameters [common]

Each object represents...

### 2.38 PendingExam [common]

Each object represents a proctored exam in progress. The existence of this record prevents starting a new proctored exam.

**serialNumber** Long The serial number of the exam.

**examId** String The exam ID.

**studentId** String The student ID.

**whenStarted** LocalDateTime The date/time the exam was started.

**startTimeMin** Integer The start time, in minutes since midnight.

**courseId** String The course ID.

**unit** Integer The unit.

**examType** EExamType The exam type:

SKILLS\_REV

UNIT

UNIT\_REV

FINAL

LICENSING

PLACEMENT

CHALLENGE

**timelimitFactor** Double The time limit factor.

**roleId** String The role under which the exam is being taken.

### 2.39 PlacementExam [common]

Each object is a record of a fee assessed to a student for a placement exam.

### 2.40 PlacementFee [common]

Each object is a record of a fee assessed to a student for a placement exam.

**studentId** String The ID of the student to whom the fee was assessed.

**courseId** String The course ID associated with math placement.

**examDate** LocalDate The date when the exam was taken.

**billDate** LocalDate The date when the fee was billed to the student account.

### 2.41 PlacementLog [common]

Each object represents a record of a placement attempt.

**serialNumber** Long The serial number.

**studentId** String The student ID.

**courseId** String The course ID.

**examId** String The exam ID.

**whenStarted** LocalDateTime The date/time when the exam was started.

**startTimeMin** Integer The start time as a minute after midnight.

**whenFinished** LocalDateTime The date/time when the exam was finished.

**finishTimeMin** Integer The finish time as a minute after midnight.

**whenRecovered** LocalDateTime The date/time when the exam was recovered.

**academicYear** String The academic year, like "1920".

### 2.42 PlacementScoreQueue [common]

Each object represents ...

### 2.43 Prerequisite [common]

Each object represents a course that can satisfy the prerequisites for a course.

**term** TermKey The term.

**courseId** String The course ID.

**prerequisite** String The ID of a course that can fulfil the prerequisite.

### 2.44 PreviousAppeals [term]

Each object represents ...

### 2.45 PreviousStudentMilestone [term]

Each object represents ...

### 2.46 PreviousStudentTerm [term]

Each object represents ...

### 2.47 Profile [common]

Each object represents a profile that a student can respond to (for example, the survey at the beginning of the placement exam, or the math plan).

**profileId** String The profile ID.

**title** String The profile title.

### 2.48 ProfileQuestion [common]

Each object represents a question within a profile.

**term** TermKey The term.

**profileId** String The profile ID.

**questionNumber** Integer The question number.

**description** String The description.

**questionType** String The question type.

**courseId** String The course ID.

**mandatory** Boolean Flag indicating question must be answered.

**questionHtml** String The question HTML.

### 2.49 ProfileQuestionChoice [common]

Each object represents a choice for a multiple-choice question within a profile.

**term** TermKey The term.

**profileId** String The profile ID.

**questionNumber** Integer The question number.

**choiceNumber** Integer The choice number.

**choice** String The choice.

**description** String The description.

**choiceMeaning** String What it means if the student selects the choice.

**choiceHtml** String The choice HTML.

### 2.50 RequiredPrereq [common]

Each object represents a prerequisite that is required to register for a course.

**course** String The course ID.

**prereqCourses** List<String> The prerequisite courses.

**mayBeConcurrent** Boolean The prerequisite may be taken concurrently.

**prereqGrades** List<String> The prerequisite grades (null for courses with no grade requirement).

### 2.51 Resource [common]

Each object represents a resource that can be lent to a student.

**resourceId** String The resource ID.

**resourceType** String The resource type.

**resourceDesc** String The description

**daysAllowed** Integer The number of days allowed before a return of the item is considered late.

**holdsAllowed** Integer The maximum number of holds allowed.

**holdId** Integer The ID of the hold to apply if the resource is not returned.

### 2.52 RuleSet [term]

Each object represents a rule set that a course section can be assigned to use.

NOTE: There must be only one rule set (pacing structure) in a term that has re\_due\_date\_enforced = 'Y'. That record is used to select registrations for which review exam points are to be accumulated during final grading.

**term** TermKey The term.

**ruleSetId** String The rule set ID.

**ruleSetName** String? The rule set name.

**defaultPaceTrack** String? The default pace track for students working under the rule set.

**requireLicensed** Boolean True to require students to pass licensing exam to access the course.

**maxCourses** Integer The maximum number of courses a student may have in a term.

**maxOpenCourses** Integer The maximum number of courses a student may have open at a time.

**requireUnitExams** Boolean Flag indicating unit exams are required.

**scheduleSource** String The schedule source.

**isFirstObjectiveAvailable** Boolean True if first objective is always available.

### 2.53 RuleSetRule [term]

Each object represents a rule within a rule set.

**term** TermKey The term.

**ruleSetId** String The rule set ID.

**activityType** ERuleSetActivity The activity:  
 LECTURE

HOMEWORK

SR\_EXAM

UNIT\_REV\_EXAM

UNIT\_EXAM

FINAL\_EXAM

**requirement** ERuleSetPrereq The prerequisite:

LECT\_VIEWED - Lecture viewed

HW\_ATMT - Homework attempted

HW\_PASS - Homework passed

HW\_MSTR - Homework mastered

SR\_ATMT - Skills Review Exam attempted

SR\_PASS - Skills Review Exam passed

SR\_MSTR - Skills Review Exam mastered

UR\_ATMT - Unit Review Exam attempted

UR\_PASS - Unit Review Exam passed

UR\_MSTR - Unit Review Exam mastered

UE\_ATMT - Unit Exam attempted

UE\_PASS - Unit Exam passed

UE\_MSTR - Unit Exam mastered

TE\_ATMT - Unit Terminal Exam attempted

TE\_PASS - Unit Terminal Exam completed

TE\_MSTR - Unit Terminal Exam mastered

### 2.54 Student [common]

Each object represents a student.

**studentId** String The student ID.

**internalId** Integer The internal ID of the student.

**lastName** String The student's last name.

**firstName** String The student's first name.

**prefName** String The student's preferred first name.

**middleInitial** String The student's middle initial.

**screenName** String The student's screen name.

**applicationTerm** TermKey The application term.

**classLevel** String The student's class level.

**college** String The student's home college.

**department** String The student's home department.

**program** String The student's program of study.

**minor** String The student's minor.

**anticGradTerm** TermKey The student's anticipated graduation term.

**numTransferCredits** String The number of transfer credits.

**highSchoolCode** String The student's high school code.

**highSchoolGpa** String The student's GPA from high school.

**highSchoolClassRank** Integer The student's class rank in high school.

**highSchoolClassSize** Integer The student's class size in high school.

**actScore** Integer The student's ACT math score.

**satScore** Integer The student's SAT math score.

**apScore** Integer The student's AP math score.

**residency** String The student's residency.

**birthDate** LocalDate The student's birth date.

**gender** String The student's gender.

**holdSeverity** String The severity of the most severe hold on the student account.

**timelimitFactor** Double The time limit factor for exams.

**isLicensed** Boolean True if the student has passed the licensing exam.

**campus** String The campus to which the student is assigned.

**email** String The student's email address.

**adviserEmail** String The student's adviser's email address.

**admitType** String The admission type.

**courseOrder** String The recommended course order (based on calculus course intentions).

**ruleSetId** String The rule set ID under which the student is working.

**admitted** Boolean True if student is admitted.

**whenCreated** LocalDateTime The date/time the student record was created.

**isRealStudent** Boolean True for real students; false for test students.

### 2.55 StudentAccommodation [common]

Each object represents a record of an accommodation letter for a student.

**studentId** String The student ID.

**term** TermKey The term in which the accommodation was received.

**whenCreated** LocalDateTime The date/time the accommodation was entered.

**accommodations** String A description of the accommodations.

**comment** String Comments.

**interviewer** String The person who entered the accommodation.

### 2.56 StudentAppeal [term]

Each object represents a record of an appeal by a student for a change to a milestone deadline.

**studentId** String The student ID.

**term** TermKey The term in which the accommodation was received.

**whenCreated** LocalDateTime The date/time the appeal was made.

**reliefGiven** Boolean True if deadline was changed as a result of the appeal.

**pace** Integer The student's pace (number of courses).

**paceTrack** String The student's pace track.

**milestoneNumber** Integer The milestone number (3-digit: pace / course / unit)

**milestoneType** EMilestoneType The milestone type being appealed:

SKILLS\_REVIEW

UNIT\_REVIEW\_EXAM

UNIT\_EXAM

FINAL\_EXAM

FINAL\_LASTTRY

**milestoneDate** LocalDate The original milestone date.

**newDeadlineDate** LocalDate The new deadline date, if relief was given.

**attemptsAllowed** Integer The number of attempts allowed (for F1 milestones)

**circumstances** String The circumstances of the appeal.

**comment** String Comments.

**interviewer** String The person who entered the accommodation.

### 2.57 StudentCategory [common]

Each object represents the assignment of a student to a special category.

**studentId** String The student ID.

**specialCategory** String The category.

**startDate** LocalDate The first date the student will belong to the category (null if none).

**endDate** LocalDate The last date the student will belong to the category (null if none).

### 2.58 StudentChallengeAttempt [common]

Each object represents an attempt on a course challenge exam by a student.

**term** TermKey The term in which the attempt took place

**serialNumber** Long The attempt serial number.

**studentId** String The student ID.

**examId** String The exam ID.

**revision** Integer The revision number.

**instance** Integer The instance number.

**whenStarted** LocalDateTime The date/time the attempt was started.

**startTimeMin** Integer The start time, in minutes since midnight.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**examLabel** String The exam label.

**courseId** String The ID of the challenged course.

**sequenceNumber** Integer The sequence number of the attempt.

**academicYear** String The academic year, like "1920".

**lastName** String The student's last name.

**firstName** String The student's first name.

**middleInitial** String The student's middle initial.

**score** Integer The student's score.

**passed** String "Y" if the student passed; "N" if not.

**howValidated** String How results were validated ("P" if proctored)

### 2.59 StudentChallengeAttemptAnswer [common]

Each object represents a student's answer to one question within an attempt on a course challenge exam.

**studentId** String The student ID.

**courseId** String The ID of the challenged course.

**examId** String The exam ID.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**questionNumber** Integer The question number.

**answer** String The student's answer.

**correct** Boolean True if the answer was correct.

### 2.60 StudentCourse [term]

Each object represents a record of a student's registration in a course during the current semester.

**term** TermKey The term.

**studentId** String The student ID.

**courseId** String The ID of the course in which the student is registered.

**sectionNum** String The section number of the course in which the student is registered.

**paceOrder** Integer The pace order of this course.

**openStatus** EOpenStatus The open status:

DROPPED

FORFEIT

NOT\_OPEN

OPEN

**gradingOption** String The selected grading option.

**isCompleted** Boolean True of course is completed.

**courseScore** Integer The total score (populated when course completed).

**courseGrade** String The course grade.

**courseGradeGpa** Float The course grade as a GPA value, only if passing.

**prereqSatisified** EProvisionalBoolean True if the prerequisites are satisfied.

**onInitialClassRoll** Boolean True if student was on initial class roll.

**onFinalClassRoll** Boolean True if student was on final class roll.

**placedByExamType** String Type of exam by which student earned registration.

**zeroUnit** Integer Unit number in which student is assigned a zero score.

**instructionType** EInstructionType The instruction type:

AP - Advance placement

OT - Credit by challenge exam

RI - Resident instruction

CE - Continuing education

CR - Continuous registration

CT - Unknown code

**registrationStatus** String The student's registration status.

**lastClassRollDate** LocalDate The last date new registrations can be created.

**isProvisional** Boolean True if registration is provisional.

**isSynthetic** Boolean True if registration is synthetic (not from a database record).

### 2.61 StudentCourseUnit [term]

Each object represents the status of a student in one unit of a course in which they are registered. These records are created when a new registration is created - there should be N + 1 records for every active registration, where N is the number of units in the course (the addition record representing unit 0). They are updated whenever an exam is submitted or when a milestone deadline is overridden for a student, and provide a quick way to display course status.

**term** TermKey The term.

**studentId** String The student ID.

**courseId** String The ID of the course in which the student is registered.

**unit** Integer The unit.

**reviewExamStatus** EStudentExamStatus The student's status with respect to the unit review exam:

NOT\_APPLICABLE (for a final exam unit with no review exam)

NOT\_ATTEMPTED

NOT\_PASSED

PASSED\_ON\_TIME

PASSED\_LATE

**reviewExamScore** Integer The best score so far on the review exam (null if not attempted).

**reviewExamPoints** Integer The points awarded toward course grade for the review exam.

**proctoredExamStatus** EStudentExamStatus The student's status with respect to the proctored unit/final exam:  
 NOT\_APPLICABLE (for a skills review unit with no proctored exam)

NOT\_ATTEMPTED

NOT\_PASSED

PASSED\_ON\_TIME

PASSED\_LATE

**proctoredExamScore** Integer The best score so far on the proctored exam (null if not attempted).

**proctoredExamPoints** Integer The points awarded toward course grade for the proctored exam.

### 2.62 StudentDiscipline [common]

Each object represents a record of a disciplinary incident with a student.

**studentId** String The student ID.

**courseId** String The course ID.

**unit** Integer The unit.

**actionType** String The type of action taken.

### 2.63 StudentEText [common]

Each object represents a record of an e-text purchased by a student.

**studentId** String The student ID.

**etextId** String The e-text ID.

**etextKey** String The e-text key.

**whenActive** LocalDate The date the key was activated.

**expirationDate** LocalDate The date the key will expire.

**refundDeadlineDate** LocalDate The deadline date for a refund.

**whenRefunded** LocalDate The date the key was refunded.

**refundReason** String The reason for the refund.

### 2.64 StudentExam [term]

Each object represents a course or tutorial exam completed by a student.

**term** TermKey The term in which the attempt took place

**serialNumber** Long The attempt serial number.

**studentId** String The student ID.

**examId** String The exam ID.

**revision** Integer The revision number.

**instance** Integer The instance number.

**whenStarted** LocalDateTime The date/time the attempt was started.

**startTimeMin** Integer The start time, in minutes since midnight.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**examLabel** String The exam label.

**courseId** String The course ID.

**sequenceNumber** Integer The sequence number of the attempt.

**examScore** Integer The student's score.

**timeOk** Boolean True if exam was completed within time limit.

**passed** String "Y" if the student passed; "N" if not.

**isFirstPassed** BooleanTrue if this is the first passing attempt on the exam.

**unit** Integer The unit.

**examType** EExamType The exam type:

SKILLS\_REV

UNIT

UNIT\_REV

FINAL

LICENSING

PLACEMENT

CHALLENGE

**examSource** String The exam source.

### 2.65 StudentExamAnswer [term]

Each object represents a student's answer to one question within an attempt on a course or tutorial exam.

**serialNumber** Long The serial number of the exam.

**questionNumber** Integer The question number.

**answerIndex** Integer The answer index, for multiple-answer questions.

**objective** Integer The objective.

**answer** String The student's answer.

**studentId** String The student ID.

**examId** String The exam ID.

**correct** Boolean True if the answer was correct.

**whenFinished** LocalDate The date when the exam was finished.

**finishTimeMin** Integer The time the exam was finished, as minute past midnight.

**subtest** String The subtest.

### 2.66 StudentHold [common]

Each object represents a hold applied to a student's account.

**studentId** String The student ID.

**holdId** String The hold ID (see **HoldType** for definitions)

**holdSeverity** String The hold severity ("F" if fatal, "N" if nonfatal).

**whenAdded** LocalDate The date the hold was added.

**timesApplied** Integer The number of times this hold has been applied to a student record for a   
 specific resource loan or rental (from the **StudentResourceLoan** table row  
 at the time this hold is added). When this reaches the maximum defined  
 in the corresponding **Resource** record, office staff may no longer delete  
 the hold, and the student must speak with a director.

**messageAdmin** String The message to display to staff/administrators.

**messageStudent** String The message to display to the student.

### 2.67 StudentHomework [term]

Each object represents a homework set completed by a student.

**serialNumber** Long The attempt serial number.

**studentId** String The student ID.

**homeworkId** String The homework ID.

**revision** Integer The revision number.

**instance** Integer The instance number.

**whenStarted** LocalDateTime The date/time the attempt was started.

**startTimeMin** Integer The start time, in minutes since midnight.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**homeworkLabel** String The homework set label.

**courseId** String The course ID.

**sectionNumber** String The section number

**unit** Integer The unit.

**objective** Integer The objective.

**homeworkType** EHomeworkType The homework type:

HOMEWORK

LAB

DISCUSSION

**homeworkScore** Integer The student's score.

**timeOk** Boolean True if exam was completed within time limit.

**passed** String "Y" if the student passed; "N" if not.

### 2.68 StudentHomeworkAnswer [term]

Each object represents a student's answer to one question within an attempt on a homework set.

**serialNumber** Long The serial number of the homework attempt.

**questionNumber** Integer The question number.

**answerIndex** Integer The answer index, for multiple-answer questions.

**objective** Integer The objective.

**answer** String The student's answer.

**studentId** String The student ID.

**homeworkId** String The homework ID.

**correct** Boolean True if the answer was correct.

**whenFinished** LocalDate The date when the homework was finished.

**finishTimeMin** Integer The time the homework was finished, as minute past midnight.

### 2.69 StudentIncomplete [term]

Each object represents an incomplete that the student has in the current term, for a course that was started in a prior term.

**term** TermKey The term.

**studentId** String The student ID.

**courseId** String The ID of the course in which the student is registered.

**sectionNum** String The section number of the course in which the student is registered.

**incInProgress** Boolean True if this is an incomplete from a prior term.

**incCountForPace** Boolean True if the incomplete is counted toward student's pace.

**incTermName** TermKey The term when the incomplete was earned.

**incDeadlineDate** LocalDate The deadline by which to complete the incomplete.

### 2.70 StudentLesson [term]

Each object represents status information for a student within a lesson.

**studentId** String The student ID.

**courseId** String The course ID.

**unit** Integer The unit.

**objective** Integer The objective.

**lessonId** String The lesson ID.

**lastComponentFinished** Integer The index of the last component finished.

**whenLectureViewed** LocalDateTime The date/time the student viewed the lecture.

### 2.71 StudentLicensingAttempt [term]

Each object represents an attempt on a licensing (user's guide) exam by a student.

#### 2.71.1 Member Fields

**term** TermKey The term in which the attempt took place

**studentId** String The student ID.

**examId** String The exam ID.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**whenStarted** LocalDateTime? The date/time the attempt was started.

**startTimeMin** Integer? The start time, in minutes since midnight.

**serialNumber** Long The attempt serial number, which provides a unique numeric name  
 (monotinically increasing) for a directory in the filesystem where  
 exam-related files can be stored.

**examScore** Integer The student's score.

**passed** String 'Y' if attempt was passed; 'N' if not.

**calculusCourseId** String?The calculus course the student plans to take.

**examLabel** String The exam label (derived from the finish date and time, of the form "Exam  
 Title taken MM/DD/YYYY at HH:MM am");

#### 2.71.2 Backing Database Table

Each record in the backing table represents a single **StudentLicensingAttempt** data object.

**Current table name: math.users**

**Proposed name:**  **[term].stusers**

**Fields:**

**term**  char(2), not null  
 The term name compnent of the **term** field. One of 'FA', 'SP', or 'SM'.  
 Proposed that this field is removed once the table is in the term database.

**term\_yr**  smallint, not null  
 The year compnent of the **term** field. Stored as short year - if 70 or greater, add 1900; if less  
 than 70, add 2000.  
 Proposed that this field is removed once the table is in the term database.

**stu\_id**  char(9), not null  
 The **studentId** field.

**version**  char(5), not null  
 The **examId** field.  
 Proposed field name: exam\_id with type char(12).

**exam\_dt**  date  
 The date component of the **whenFinished** field.

**finish\_tm\_s**  smallint  
 The time component of the **whenFinished** field, in seconds past midnight.  
 Proposed field.

**start\_dt**  date  
 The date component of the **whenStarted** field.  
 Proposed field.

**start\_tm\_s**  smallint  
 The time component of the **whenStarted** field, in seconds past midnight.  
 Proposed field.

**serial\_nbr**  integer  
 The **serialNumber** field.  
 Proposed that this field is "not null".

**exam\_score**  smallint, not null  
 The **examScore** field.

**passed**  char(1), not null  
 The **passed** field, "Y" if passed, "N" if not passed.

**calc\_course**  char(2)  
 The **calculusCourseId** field ('41' for MATH 141, '55' for MATH 155, '56' for MATH 156, or  
 '60' for MATH 160, null if not stated of if the exam does not gather that data.

**exam\_name**  varchar(32)  
 The name used to compose the examLabel field. From the Exam table - a convenienve field to   
 avoid having to query Exam table to get name in order to generate label.  
 Proposed field.

#### 2.71.3 Term Rollover

In the current database, all records are purged at the end of each term (a copy is retained in the term arrchive database).

In the proposed database, records are accumulated during the term in the term database, and remain in the term database as an archival record.

#### 2.71.4 API

The API is exposed via the "StudentLicensingAttemptCache" utility class.

List<**StudentLicensingAttempt**> **queryAll**();  
 Retrieves all records in the database for the active term.

List<**StudentLicensingAttempt**> **queryByStudent**(String studentId);  
 Retrieves all records in the database for the active term belonging to a specific student.

boolean **recordAttempt**(**StudentLicensingAttempt** record);  
 Records a licensing exam attempt. This inserts the attempt record, and if the attempt was passed and the "licensed" field of the student's record is not 'Y', the 'licensed' field is updated to 'Y' on the student record.

### 2.72 StudentMathPlan [common]

Each object represents a response to a math plan question by a student.

**sessionTag** Long The session under which the record was created.

**studentId** String The student ID.

**internalId** Integer The internal ID of the student.

**pageId** String The page ID containing the question.

**questionNumber** Integer The question number.

**whenSubmitted** LocalDateTime The date/time the response was submitted.

**answer** String The student's answer.

**finishTimeMin** Integer The finish time, in minutes since midnight.

### 2.73 StudentMessage [term]

Each object represents a message sent to a student.

**studentId** String The student ID.

**messageDate** LocalDate The date the message was sent.

**touchPoint** ETouchPoint The touch point at which the message is sent:

TP\_1R1 - Course 1, prior to Review Exam 1 deadline

TP\_1R3 - Course 1, prior to Review Exam 3 deadline

TP\_1FE - Course 1, prior to Final Exam deadline

TP\_1F1 - Course 1, after Final Exam deadline, before F1 deadline

TP\_1BL - Course 1, after F1 deadline

TP\_2R1 - Course 2, prior to Review Exam 1 deadline

TP\_2R3 - Course 2, prior to Review Exam 3 deadline

TP\_2FE - Course 2, prior to Final Exam deadline

TP\_2F1 - Course 2, after Final Exam deadline, before F1 deadline

TP\_2BL - Course 2, after F1 deadline

TP\_3R3 - Course 3, prior to Review Exam 3 deadline

TP\_3FE - Course 3, prior to Final Exam deadline

TP\_3F1 - Course 3, after Final Exam deadline, before F1 deadline

TP\_3BL - Course 3, after F1 deadline

TP\_4R3 - Course 4, prior to Review Exam 3 deadline

TP\_4FE - Course 4, prior to Final Exam deadline

TP\_4F1 - Course 4, after Final Exam deadline, before F1 deadline

TP\_4BL - Course 4, after F1 deadline

TP\_5R3 - Course 5, prior to Review Exam 3 deadline

TP\_5FE - Course 5, prior to Final Exam deadline

TP\_5F1 - Course 5, after Final Exam deadline, before F1 deadline

**messageCode** String The message code.

**sender** String The name of the person sending the message.

### 2.74 StudentMilestone [term]

Each object represents an override to a milestone date for a student.

**term** TermKey The term.

**studentId** String The student ID.

**paceTrack** String The pace track.

**milestoneNumber** Integer The milestone number (3-digit: pace/course/unit)

**milestoneType** EMilestoneType The milestone type:

SKILLS\_REVIEW

UNIT\_REVIEW\_EXAM

UNIT\_EXAM

FINAL\_EXAM

FINAL\_LASTTRY

**milestoneDate** LocalDate The milestone date.

**attemptsAllowed** Integer The number of attempts allowed; null if unlimited.

**courseIndex** Integer The course index (1 for first course, etc.).

**unit** Integer The unit.

### 2.75 StudentPaceSummary [common]

Each object represents...

### 2.76 StudentPlacementAttempt [common]

Each object represents an override to a milestone date for a student.

**term** TermKey The term in which the attempt took place

**serialNumber** Long The attempt serial number.

**studentId** String The student ID.

**examId** String The exam ID.

**revision** Integer The revision number.

**instance** Integer The instance number.

**whenStarted** LocalDateTime The date/time the attempt was started.

**startTimeMin** Integer The start time, in minutes since midnight.

**whenFinished** LocalDateTime The date/time the attempt was finished.

**finishTimeMin** Integer The finish time, in minutes since midnight.

**examLabel** String The exam label.

**courseId** String The course ID.

**sequenceNumber** IntegerThe sequence number.

**academicYear** String The academic year.

**lastName** String The student's last name.

**firstName** String The student's first name.

**middleInitial** String The student's middle initial.

**placed** String "Y" if student placed, "N" if not

**howValidated** String The method by which the result was validated

**subtestScores** Map<String, Integer> Map from subtest name to score.

### 2.77 StudentPlacementAttemptAnswer [common]

Each object represents a student's answer to one question within a placement exam attempt.

**serialNumber** Long The attempt serial number.

**studentId** String The student ID.

**examId** String The exam ID.

**whenFinished** LocalDate The date when the exam was finished.

**finishTimeMin** Integer The time the exam was finished, as minute past midnight.

**questionNumber** Integer The question number.

**answer** String The student's answer.

**correct** Boolean True if the answer was correct.

**subtest** String The subtest.

**treeRef** String The tree reference of the selected item.

### 2.78 StudentPlacementCredit [common]

Each object represents placement into or credit for in a single course on a placement or challenge attempt.

**studentId** String The student ID.

**courseId** String The course ID.

**placementResult** String The placement result ("P" for placement, "C" for credit)

**serialNumber** Long The serial number of the placement attempt.

**examId** String The exam ID.

**examSource** String The exam source.

**whenFinished** LocalDate The date of the placement attempt.

**whenCreditRefused** LocalDate The date the student refused placement credit.

### 2.79 StudentPlacementDenied [common]

Each object represents a placement or credit result that was denied.

**studentId** String The student ID.

**courseId** String The course ID.

**placementResult** String The placement result ("P" for placement, "C" for credit)

**serialNumber** Long The serial number of the placement attempt.

**examId** String The exam ID.

**examSource** String The exam source.

**whenFinished** LocalDate The date of the placement attempt.

**whyDenied** String The reason the result was denied.

### 2.80 StudentProfileResponse [common]

Each object represents a student response to a profile question.

**studentId** String The student ID.

**profileId** String The profile ID.

**questionNumber** Integer The question number.

**whenSubmitted** LocalDateTime The date/time the response was submitted.

**answer** String The answer.

**finishTimeMin** Integer The submission time, in minutes past midnight.

### 2.81 StudentResourceLoan [common]

Each object represents a student response to a profile question.

**studentId** String The student ID.

**resourceId** String The ID of the resource that was lent.

**loanDateTime** LocalDateTime The date/time the resource was lent.

**dueDate** LocalDate The due date.

**returnDateTime** LocalDateTime The date/time the resource was returned.

**timesDisplayed** Integer The number of times the student has been shown overdue notice.

### 2.82 StudentTerm [term]

Each object represents a student's status within a term.

NOTE: When doing final grading for Incompletes, the record from the term where the incomplete was started is used to determine the pace and track.

**term** TermKey The term.

**studentId** String The student ID.

**pace** Integer The student's pace.

**paceTrack** String The student's pace track

**firstCourseId** String The ID of the student's first course.

### 2.83 StudentTransferCredit [common]

Each object represents a course for which the student has receive transfer credit.

**studentId** String The student ID.

**courseId** String The course ID.

**numCredits** Double The number of credits.

**grade** String The transferred grade.

**gradeGpa** Double The transferred grade as a GPA, only if passing.

**whenAdded** LocalDate The date when the record was added.

**whenCreditRefused** LocalDate The date transfer credit was refused.

### 2.84 StudentVisit [common]

Each object represents a record of a student visiting an area of the center.

**studentId** String The student ID.

**whenEntered** LocalDateTime The date/time when the student entered the area.  
**whenExited** LocalDateTime The date/time when the student exited the area.  
**location** ECenterArea A 2-character code indicating the area visited.

**seat** String The seat number, where seating is tracked.

### 2.86 TermWeek [term]

Each object represents the definition of a week within a term.

**term** TermKey The term.

**weekInTerm** Integer The week in the term.

**startDate** LocalDate The week start date.

**endDate** LocalDate The week end date.

### 2.87 TestingCalculator [common]

Each object represents a record of a calculator lent to a student in the testing center.

**studentId** String The student ID.

**calculatorNumber** String The ID of the calculator that was lent.

**examSerialNumber** Long The serial number of the associated exam.

**examDate** LocalDate The exam date.

**whenIssued** LocalTime The time the calculator was issued.

**whenReturned** LocalTime The time the calculator was returned.

### 2.88 TestingCenter [common]

Each object represents a testing center.

**testingCenterId** String The testing center ID.

**name** String The testing center name.

**whenCreated** LocalDateTime The date/time the record was created.

**whenApproved** LocalDateTime The date/time the record was approved.

**remote** Boolean True if center is remote.

**proctored** Boolean True if center is proctored.

### 2.89 TestingStation [common]

Each object represents a testing station.

**computerId** String The testing station ID.

**testingCenterId** String The testing center ID.

**stationNumber** String The station number.

**description** String The description.

**iconX** Integer The X coordinate of the icon.

**iconY** Integer The Y coordinate of the icon.

**computerUsage** EComputerUsage The computer usage:

ONLINE

PAPER

BOTH

DISABLED

WHEELCHAIR

**computerState** EComputerState The current state of the computer:

ERROR

UNINITIALIZED

CANT\_CONNECT

GETTING\_INFO

CANT\_GET\_INFO

LOCKED

PAPER\_ONLY

AWAIT\_STUDENT

TAKING\_EXAM

EXAM\_RESULTS

FORCE\_SUBMIT

WAIT\_CHECKOUT

DELETE\_CERTS

CANCEL\_EXAM

LOGIN\_NOCHECK

LOGIN\_COUPONS

**whenCreated** LocalDateTime The date/time the record was created.

**whenApproved** LocalDateTime The date/time the record was approved.

**currentStudentId** String The ID of the student currently assigned to the station.

**currentCourseId** String The ID of the course of the exam currently in progress.

**currentUnit** Integer The unit of the exam currently in progress.

**currentExamId** String The ID of the exam currently in progress.

### 2.90 UngradedSection [term]

Each object represents a section whose grades should not be included in end-of-term grade submission (because a face-to-face instructor will submit them manually).

**courseId** String The course ID.

**sectionNumber** String The section number.

**term** TermKey The term.

### 2.91 UnproctoredPlacement [term]

Each object represents a date range when incoming students may complete the math placement tool without proctring.

**term** TermKey The term.

**applicationTerm** TermKey The application term of students who can use this date range.

**courseId** String The course ID.

**startDate** LocalDate? The start date.

**endDate** LocalDate? The end date.

### 2.92 UserPermission [term]

Each object represents...

### 2.93 ZipCode [common]

Each object represents a zip code.

#### 2.93.1 Member Fields

**zipCode** String The zip code, trimmed to 10 characters in length.

**city** String The city, trimmed to 18 characters in length.

**state** String The state's 2-letter code (uppercase).

#### 2.93.2 Backing Database Table

Each record in the backing table represents a single **Calendar** data object.

**Current table name: math.zip\_code**

**Proposed name:**  **[common].zip\_code**

**Fields:**

**zip\_code** char(10), not null  
 The **zipCode** field.

**city** char(18), not null  
 The **city** field

**state** char(2), not null  
 The **state** field.

#### 2.93.3 Term Rollover

This data is retained indefinitely, and periodically updated. It is used for highschool reporting, so it only needs to include zip codes for high schools that serve Colorado.

#### 2.93.4 API

The API is exposed via the "ZipCodeCache" utility class.

List<ZipCode> queryAll();

ZipCode query(String zipCode);

### 2.3. Analytics Schema

#### 2.3.1 AnalyticsStudent

Each object represents a single student for which analytics data is present. This can include students who have taken Math courses, or those who have used the Math Placement process. They need not be a current student. Typically, this will be the set of all students retained in the current "student" or "applicant" tables.

Estimated size: 400,000 records.

##### 2.3.1.1 Member Fields

**pidm** Integer The PIDM of the student

**stu\_id** String The CSU ID of the student

**last\_name** String The student's last name (upper case)

**first\_name** String The student's first name (upper case)

**birth\_date** date The student's birth date

**gpa** number The student's current CSU GPA

**sex** char(1) The student's sex (M, F, U)

**hisp\_lat\_race** char(1) "Y" if student identifies as Hispanic/Latino

**amer\_ind\_race** char(1) "Y" if student identifies as American Indian

**asian\_race** char(1) "Y" if student identifies as Asian

**black\_race** char(1) "Y" if student identifies as Black

**hawaiian\_race** char(1) "Y" if student identifies as Hawaiian

**white\_race** char(1) "Y" if student identifies as White

**multi\_race** char(1) "Y" if student identifies as Multi-racial

**apln\_term** Integer The student's application term, in "202310" format.

**admit\_term** Integer The term the student was admitted (null if not yet admitted).

**admit\_program** char(14) The program of study into which student was admitted

**hs\_code** char(6) The student's high-school code

**hs\_grad\_date** date The student's high-school graduation date

**hs\_gpa** number The student's high-school GPA

**hs\_class\_rank** integer The student's high-school class rank

**hs\_class\_size** integer The student's high-school class size

**act\_math** integer The student's maximum ACT Math score

**sat\_math** integer The student's maximum SAT Math score

**satr\_math** integer The student's maximum SAT (Revised) Math score

**status** Integer 1 = applicant  
 2 = admitted  
 3 = matriculated, active student  
 4 = graduated  
 5 = planned leave  
 6 = dropped out

##### 2.3.1.2 Schema Table

Each record in the backing table represents a single **AnalyticsStudent** data object.

**Proposed name:**  **[anlyt].student and [anlyt\_t].student**

**SQL:**

CREATE TABLE IF NOT EXISTS anlyt.student (  
 **pidm** integer not null, -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.PIDM

**stu\_id** char(9) not null, -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.CSU\_ID

**last\_name** char(30) not null, -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.LAST\_NAME

**first\_name** char(30) not null, -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.FIRST\_NAME

**birth\_date** date, -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.BIRTH\_DATE

**gpa** numeric(4,2), -- CSUBAN.CSUG\_GP\_ACADEMIC\_UG.GPA

**sex** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.GENDER

**hisp\_lat\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.HISPANIC\_LATINO\_ETHNICITY\_IND

**amer\_ind\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.AMERICAN\_INDIAN\_RACE\_IND

**asian\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.ASIAN\_RACE\_IND

**black\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.BLACK\_RACE\_IND

**hawaiian\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.HAWAIIAN\_RACE\_IND

**white\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.WHITE\_RACE\_IND

**multi\_race** char(1), -- CSUBAN.CSUG\_GP\_ADMISSIONS.MULTI\_RACE\_IND

**apln\_term** integer, -- CSUBAN.CSUS\_APPLICANT.TERM

**admit\_type** char(2), -- CSUBAN.CSUS\_APPLICANT.ADMIT\_TYPE

**admit\_term** integer, -- (derived from CSUBAN.CSUS\_APPLICANT.INST\_LATEST\_DECN)

**admit\_program** char(14), -- CSUBAN.CSUS\_APPLICANT.ADM\_PROGRAM\_OF\_STUDY

**hs\_code** char(6), -- CSUBAN.CSUG\_GP\_ADMISSIONS.HS\_CODE

**hs\_grad\_date** date, -- CSUBAN.CSUG\_GP\_ADMISSIONS.HS\_GRAD\_DATE

**hs\_gpa** numeric(4,2), -- CSUBAN.CSUG\_GP\_ADMISSIONS.HS\_GPA

**hs\_class\_rank** integer, -- CSUBAN.CSUG\_GP\_ADMISSIONS.HS\_CLASS\_RANK

**hs\_class\_size** integer, -- CSUBAN.CSUG\_GP\_ADMISSIONS.HS\_CLASS\_SIZE

**act\_math** integer, -- CSUBAN.CSUS\_TEST\_SUPER\_SCORES.ACT\_A02\_MAX

**sat\_math** integer, -- CSUBAN.CSUS\_TEST\_SUPER\_SCORES.SAT\_S02\_MAX

**satr\_math** integer, -- CSUBAN.CSUS\_TEST\_SUPER\_SCORES.SATR\_S12\_MAX

**status** integer -- (derived from CSUBAN.CSUS\_APPLICANT.INST\_LATEST\_DECN)

) TABLESPACE anlyt\_tbl;

CREATE TABLE IF NOT EXISTS anlyt\_t.student (**...**) TABLESPACE test;

##### 2.3.1.3 Indexes

CREATE UNIQUE INDEX IF NOT EXISTS anlyt\_i\_student\_pidm ON anlyt.student (pidm) TABLESPACE anlyt\_idx;

CREATE INDEX IF NOT EXISTS anlyt\_i\_student\_id ON anlyt.student (stu\_id) TABLESPACE anlyt\_idx;

##### 2.3.1.4 API

The API is exposed via the "AnalyticsStudent" utility class.

List<AnalyticsStudent> queryAll();

List<Integer> queryAllAplnTerms();  
 List<Integer> queryAllAdmitTerms();

List<Integer> queryAllGradTerms();

List<AnalyticsStudent> queryAllByAplnTerm(int aplnTerm);

List<AnalyticsStudent> queryAllByAdmitTerm(int admitTerm);

List<AnalyticsStudent> queryAllByGradTerm(int gradTerm);

AnalyticsStudent queryByPidm(long pidm);

AnalyticsStudent queryByStuId(String stuId);