MathOps.Dev Schemata

The MathOps system includes a wide variety of data spanning many different patterns of usage and generation.

* Configuration data such as course, section, term, and policy definitions (low-volume, frequent reads)
* Dynamic student data like registrations, course status, profiles (high-volume, frequent reads/writes)
* Records of activities like logs or messages (high-volume, frequent writes)
* Analytics data (high-volume, frequent reads)

Low-volume data that is read frequently is best served by an in-memory replicated cache that can be populated on startup from a “source of truth”. An administrative interface to alter this data would update the “source of truth” and update the in-memory data at the same time. In-memory data would be replicated across multiple nodes in a high-availability cluster.

High-volume data would be better served by a partitioned databae such as **Cassandra**, since queries in this system tend to rely on a few small key columns, and joins or database-level referential integrity constraints are not needed.

The MathOps system stores data in the context of “terms”, or semesters. For example, Fall, Spring, and Summer semesters of each academic year. There is exactly one term “active” at any moment in time. Each term represents a contiguous span of days. Terms do not overlap, and there are no gaps between terms. Old term configuration data is retained for archival and reference. New term data can be prepared ahead of time, to become active when the current date crosses the boundary into a new term.

Data for the current term will be accessed frequently (with many updates), data for older terms will be accessed infrequently (and without updates), and data for upcoming terms will be created or updated infrequently as new terms approach. Therefore, we do not want to store data for all terms in all tables and force scanning of large amounts of non-applicable data for each query within the active term. To address this, we will create a schema for every term.

At a large scale, data is partitioned into several tablespaces:

* **main** (data that does not vary by term, such as student data)
* **extern** (data that comes from "external" systems like a University registrar's database)
* **analyt** (analytics data that is updated when analytics need to be run)
* **termYYYYMM** (one tablespace for each term, where YYYY is a 4-digit year and MM is a unique 2-digit code  
  for the term within the year, where 30 indicates a Spring term, 60 a Summer term, and 90 a Fall term)

Since some database products (like Cassandra) perform best when the numbers of tables to search through is small (say, under 100), tablespaces have been divided into schemas, where each schema could run within its own cluster.

# Schema 1. **main.system**

The **system** schema, in the **main** tablespace, stores top-level system configuration.

This schema includes the following object types:

* **Term** – One object for every term configured.
* **School** – One object for each schools, such as high schools or other post-secopndary institutions with which the program may interact.
* **HoldType** – One record for each type of administrative hold that can be applied to a user account.
* **LocalLogin** – One record for every configured local login.
* **Parameters** – One record for every system-configuration parameter set.
* **UserPermission** – One record for every permission granted to a user.
* **ZipCode** – One record per zip code that a school's address may reference.

## Conceptual Data Model

### Term

Description: Each **Term** object stores the top-level configuration of a term.

When a new **Term** object is created, the corresponding tablespace (and its tables) should be created. When a **Term** object is deleted, the corresponding tablespace (and its tables) should be archived to files then deleted. Applications can query for the active term, then use that to select the appropriate term tablespace for queries of term-specific data.

At the boundary between terms (midnight at the end of a term’s last day), the “active index” of all **Term** objects should be incremented.

Example: Under a typical “fall, spring, summer” academic year, if the current year is 2023, and the Fall term is active, there would be a “Fall 2023” term with active index 0, a “Spring 2024” term with active index 1, a “Summer, 2024” term with active index 2, a “Summer 2023” term with active index -1, and so forth.

Attributes:

* Term ID (a 6-digit number YYYYMM where YYYY is the year, and MM is 30 for Spring, 60 for Summer, 90 for Fall, or other values for custom term types)
* Start date
* End date
* Academic year (a four-digit number like “2324” indicating the 2023/2024 year)
* Active index (0 for the active term, -1 for the prior term, +1 for the next term, etc.)
* Drop deadline date
* Withdraw deadline date

### School

Description: Each **School** object stores information on a single school.

Example: A student's high school may be known (based on a CIP code), and aggregate statistics for all students from that high school can be sent annually to the school to indicate how those students are doing once they reach the University.

Attributes:

* Code (a unique code for the school – the CIP code for a high school)
* Name (the name of the school, suitable for use on a mailing label)
* Address (the mailing address, suitable for use on a mailing label)
* City (the city name, suitable for use on a mailing label)
* State (the 2-letter state abbreviation, suitable for use on a mailing label)
* Zip Code (the postal code, suitable for use on a mailing label)

### HoldType

Description: Each **HoldType** object represents a type of administrative hold that can be applied to a user's account.

Example: A student's who has borrowed a resource but not returned it by its due date may receive a hold that prevents further course work until the resource is returned. Holds are generally applied by nightly automated processes, so removing a hold is a temporary fix unless the underlying cause of the hold is remediated.

Attributes:

* Hold ID (a unique ID for each hold, 2 characters in length)
* Severity (the severity of the hold, which indicates the actions the hold prevents)
* Hold type (a category for the hold, such as registration, resource, or disciplinary)
* Who can add (a code indicating who is allowed to add the hold)
* Who can delete (a code indicating who is allowed to delete the hold)
* Times to display (if non-null, the hold is informational, and should be shown to students this number of times before being automatically deleted)

### LocalLogin

Description: Each **LocalLogin** object represents a set of login credentials with which a user can log into the system. This is separate from any institutional centralized login system, and can be used for system administrators to access the system.

This can also be used to provide a way for someone not affiliated with the institution to log in.

Attributes:

* Login (a unique login name – users may also be able to use an email address to log in)
* User ID (the ID of the user)
* User type (the type of user – administrator, director, proctor, tutor, student, etc.)
* Email (the user's email address – must be unique within this table)
* Salt (a salt value for the SCRAM-SHA-256 algoritme, as a 32-character string)
* Stored Key (a SCRAM-SHA-256 stored key, in hexadecimal, 64 characters in length)
* Server Key (a SCRAM-SHA-256 server key, in hexadecimal, 64 characters in length)
* When created (the date/time the login was created)
* When expires (an optional expiation date/time)
* When last logged in (the date/time the user last successfully logged in)
* When last failed (the date/time the user last failed to log in)
* Number of fails (the number of failures since the last successful log in)
* Force password change (True to force the user to change their password on login)

### UserPermission

Description: Each **UserPermission** object represents a permission level for a single activity granted to a single user.

Attributes:

* User ID (the ID of the user)
* Activity (the ID of the activity)
* Permission Level

### Parameters

Description: Each **Parameters** object represents a named collection of parameter values that can control operation of system functions like batch jobs, reports, etc.

Attributes:

* Name (the name of the parameter set, to be referenced by configurable jobs)
* String parameters (a list of zero or more String parameters)
* Integer parameters (a list of zero or more Integer parameters)
* Date parameters (a list of zero or more date parameters)
* Time parameters (a list of zero or more time parameters)
* DateTime parameters (a list of zero or more date/time parameters)

### ZipCode

Description: Each **ZipCode** object stores a zip code with the associated city and state.

Example: This could be used to auto-populate an address when entering a new school.

Attributes:

* Zip Code (five digit, not the 5+4 format)
* City (the city name, suitable for use on a mailing label)
* State (the 2-letter state abbreviation, suitable for use on a mailing label)

### Term Week [→ Term]

Keyspace: **[termYYYYMM]**

Description: Each **Term Week** object stores the date ranges for a single “week” within a term. These ranges need not correspond exactly to calendar weeks, but are used when reporting in which week a student’s work occurred. The weeks of a term must cover all days contained in the term, but may not overlap (every calendar day must uniquely identify a term and a week number within that term).

Example: Week 0 could represent 12 days from the start of the term to the first day of the first week of class. Then weeks 1 through 15 would be the actual weeks of the class (with each week running from Sunday through Saturday), but week 10 could span 14 days since it covers “Spring Break”. Then a week 16 could cover 20 days from the day after the last day of classes to the end of the term.

Attributes:

* Week number
* Start date
* End date

### Pace Track [→ Term]

Keyspace: **[termYYYYMM]**

Description: Each **Pace Track** object represents a “track” to which a student can be assigned. A track organizes some number of courses sequentially within a single term and manages the due dates and milestones for each course. A student’s track depends on their course registrations in the term. This can change as the student adds or removes registrations over the course of a term. There should be a pace track for every pace with criteria of “DEFAULT”, to be used when none of the other criteria match.

Example: “Pace 2, Track A” will specify a set of milestone dates for two courses in a term. Only one record with a specified pace and track may exist – one may not re-use track names within different subterms. “Pace 2, Track B” could give different milestone dates for two courses in the same subterm. “Pace 2, Track C” could be used in a different sub-term, with milestone dates appropriate for that portion of the term.

Attributes:

* Pace (the number of courses for which the student has registered)
* Track (the name of the track, unique within each Pace)
* Subterm (a code for the portion of the term in which this track defines deadlines – subterms may overlap)
* Criteria (a rule for assigning students to this pace track based on their registrations)

### Pacing Structure [→ Term]

Keyspace: **[termYYYYMM]**

Description: Each **Pacing Structure** object represents the way courses are sequenced within a track, and may constrain the number of courses that can be taken in a single term. A student must have only a single pacing structure for each pacing group for which they have a registration. Every section of a course will define its pacing structure and pacing group ID.

Example: Typical pacing structures are “instructor led” vs. “student-managed”. In the former, milestone dates will be based on a fixed schedule, and assignments might open up for fixed windows of time. In the latter, assignments may become available as earlier work is completed. Some pacing structures would allow multiple courses to be open at the same time. Some pacing structures allow students to qualify for incompletes in courses by completing some portion of the course.

Attributes:

* Pacing Structure ID
* Title
* Maximum number of courses in a term
* Maximum number of courses open at one time
* Schedule source (pace, fixed)
* Allow Incompletes?
* Requires pacing structure licensing exam?
* Minimum units to complete to enable incomplete

### Rule Set [→ Term]

Keyspace: **[termYYYYMM]**

Description: Each **Rule Set** object represents a collection of rules that govern how courses operate. Every section of a course specifies the rule set it uses (different sections may use different rule sets). A rule set dictates what tasks the student must complete in order to access subsequent tasks, and can control other aspects of course delivery. Defining rules within a rule set allows multiple course sections to adopt identical rules without having to encode them within each course section.

Example:

Attributes:

* Rule Set ID
* Title
* Requires rule set licensing exam?

Relationships:

* 1 **Rule Set** contains N **Rule Set Rule** objects

### Rule Set Rule [→ Rule Set]

Keyspace: **[termYYYYMM]**

Description: Each **Rule Set Rule** object represents a single rule within a rule set, which specifies that some requirement must be met in order to access some activity.

Attributes:

* Activity type
* Requirement

### Course

Keyspace: **main**

Description: Each **Course** object corresponds to either a course offered in the University catalog, or to a “placeholder” course used to manage access to non-credit tutorials or placement outcomes. This table stores only courses for which content is delivered by the system. There is a separate **Catalog Course** object to store information about all courses in the catalog.

Example: There would normally be one record for each course managed by the system, plus one record for each non-credit tutorial, one for the math placement exam, and several for various placement outcomes. There may also be entries for courses that provide student “guest access” to managed courses.

Attributes:

* Course ID (like “MATH 117”, from the University catalog)
* Title (like “College Algebra I”, from the University catalog)
* Label (the course label, often identical to the course ID)
* Inline prefix (for example, “The ” if the Title is “Math Tutorial”; null or empty if no prefix needed, such as for the title “College Algebra I”).
* Catalog URL
* Number of credits
* Type (course, non-credit tutorial, placement activity, licensing exam, survey, etc.)

Relationships:

* 1 **Course** contains N **Course Prerequisite** objects
* 1 **Course** contains N **Course Variant** objects

### Course Prerequisite [→ Course]

Keyspace: **main**

Description: Each **Course Prerequisite** object defines one or more courses (possibly with required minimum grades in each), credit for all of which satisfies the prerequisite for this course. If there are multiple records for a course, having credit that matches ANY record is sufficient to clear the prerequisite for the parent course.

Example: The prerequisites for a College Algebra I course could include courses transferred from local community colleges or math placement outcomes, which are expressed in terms of course IDs.

Attributes:

* List<Prerequisite course ID>
* List<Minimum grade required>

### Course Variant [→ Course]

Keyspace: **main**

Description: Each **Course Variant** object corresponds to a variant of a single **Course** object. Variants may have differing numbers of units and standards, and may be offered in different formats, but should all have reasonable similar course learning outcomes.

Example: A course being re-designed and offered in a legacy format while a few sections of the new format are piloted would have a legacy variant and a new variant.

Attributes:

* Variant ID (such as the name of the curriculum author or a term ID if only offered one term)
* Number of units
* When active
* When pulled
* Requred E-Text ID (null if none)

Relationships:

* 1 **Course Variant** contains N **Course Unit** objects
* 1 **Course Variant** contains N **Course Section** objects within each term

### Course Unit [→ Course Variant]

Keyspace: **main**

Description: Each **Course Unit** object corresponds to either a single unit pr module of a **Course Variant** object. Every course variant has a fixed number of units.

Example: Unit 0 as a “Skills Review” unit, then 4 units of content, and a unit to provide a comprehensive final exam. Courses that represent only things like placement or licensing exams would have a single unit of “Exam” type, but which could contain instructional review content.

Attributes:

* Unit
* Title
* Type (Skills Review, Instruction, Final Exam, Exam)

Relationships:

* 1 **Course Unit** contains N **Course Unit Objective** objects

### Course Unit Objective [→ Course Unit]

Keyspace: **main**

Description: Each **Course Unit Objective** object corresponds to either a single objective or learning target/standard within a **Course Unit** object. Objectives represent a finer level of granularity in the organization of instructional content than a unit. Objectives in a unit may be presented linearly or students may have some control over ordering. In a “standards-based” course, each objective represents one standard. In a mastery-based course, each objective represents a topic.

Example: A unit might have five objectives, each one with some instructional content, a set of exercises, and an assignment.

Attributes:

* Objective ID (like “MATH 117”)
* Type (Introduction, Skills Review, Learning Target, etc.)
* Label (a label within this course unit, like “Learning Target 3.1”)
* Lesson ID (the ID of a packaged lesson that provides instructional content)

Relationships:

* 1 **Course Unit Objective** contains N **Course Unit Objective Lesson** objects

### Course Unit Objective Lesson [→ Course Unit Objective]

Keyspace: **main**

Description: Each **Course Unit Objective Lesson** object associates a lesson with a course unit objective, and configures the lesson’s role in the course.

Example: A standards-based course could have three objectives per unit, with several lessons within each objective (sub-targets within the standard’s learning target)

Attributes:

* Index (1-based index to order lessons within the objective)
* Label (a label within this course unit objective, like “Deriving and Applying the Law of Sines”, or null to use the native title from the Lesson)
* Lesson ID (the ID of a packaged lesson that provides instructional content)

### Lesson

Keyspace: **main**

Description: Each **Lesson** object represents a packaged block of instructional content with associated practice exercises and assignments. These can be assembled into courses by referencing them from **ourse Unit Objective** objects, or presented stand-alone or embedded in other courses as just-in-time review. A lesson can be composed of a sequence of components, each of which can be marked as “completed” by a student.

Example: A lesson on the derivation and application of the Law of Sines.

Attributes:

* Lesson ID
* Title (the lesson title)
* Type
* Description

Relationships:

* 1 **Lesson** contains N **Lesson Component** objects

### Lesson Component [→ Lesson]

Keyspace: **main**

Description: Each **Lesson Component** object represents a portion of a **Lesson** object’s content.

Example: One lesson might have an introductory or motivating video component, then three sets of an expository component and an associated formative assessment component, an application context component, a summary component, and finally a summative assessment component.

Attributes:

* Sequence Number
* Type
* XML Data or URL

Relationships:

* 1 **Lesson Component** contains N **Lesson Component Asset** objects

### Lesson Component Asset [→ Lesson Component]

Keyspace: **main**

Description: Each **Lesson Component Asset** object represents a one “asset” related to the lesson component, such as a video, presentation, PDF file, Word document, spreadsheet, data set, URL, etc. The status of each asset can be tracked, along with notes regarding desired changes, etc.

Example: An “introductory lecture” component might have a raw video asset, extracted audio, cleaned and leveled audio, processed video (perhaps in multiple resolutions or formats), video processor data file(s), a text transcript, VTT closed captions, a Word document and associated PDF version, a PowerPoint presentation and associated PDF version, source image files and SVG or PostScript or PDF drawings, attributions, notes, data sets, and so forth.

Attributes:

* Identifier
* Asset Type
* (probably many metadata fields could go here)
* XML Data or URL

### Course Section [→ Course Variant]

Keyspace: **[termYYYYMM]**

Description: Each **Course Section** object represents a single section of a **Course Variant** offered within a **Term**. Students enroll in sections.

Example: An algebra might have a face-to-face full-semester section, a face-to-face late-start section, three online sections, a distance section, and a special section for people with red hair.

Attributes:

* Section number
* CRN
* LMS ID
* University start date
* University end date
* First class date
* Last class date
* Exam delete date (used for tutorials, where if a student has not completed the tutorial by this date, which is typically a registration deadline, their progress is deleted and they need to restart the tutorial)
* Last add without override date
* Last add with override date
* Instruction type (advance placement, challenge credit, resident instruction, continuing education, continuous registration, unknown)
* Exam structure (unit and final exams, unit exams only, final exams only)
* Pacing group ID
* Pacing structure
* Rule set ID
* Grading system (S/U, A/B/C/U, A/B/C/D/F, etc)
* Minimum score for A/S
* Minimum score for B
* Minimum score for C
* Minimum score for D
* Counts toward maximum open?
* Delivery mode (face to face, hybrid, distance)
* Bulk grade submission? (true if this section’s grades will be bulk-submitted at the end of the term)
* Proctoring options (list of department testing center, university testing center, assistive testing center, distance testing center, human, ProctorU student-paid, ProctorU university-paid, department online)
* URL under which to find section-specific messaging for display in course pages

Relationships:

* 1 **Course Section** contains N **Course Section Unit** objects

### Course Section Unit [→ Course Section]

Keyspace: **[termYYYYMM]**

Description: Each **Course Section Unit** object provides the configuration of a unit within a **Course Section**.

Example: Each unit within a section could have a unique testing window.

Attributes:

* Unit (selects a **Course Unit** object within the course variant – the unit will already have a type, such as Introduction, Skills Review, Instruction, Midterm Exam, Final Exam, etc.)
* First test datetime (null if there are no fixed testing windows)
* Last test datetime (null if there are no fixed testing windows)
* URL under which to find section-specific messaging for display in course pages

Relationships:

* 1 **Course Section Unit** contains N **Course Section Unit Objective** objects

### Course Section Unit Objective [→ Course Section Unit]

Keyspace: **[termYYYYMM]**

Description: Each **Course Section Unit Objective** object provides the configuration of an objective within a **Course Section Unit**.

Example: Different sections may define different scoring thresholds for mastery or completion of assignments.

Attributes:

* Objective (selects a **Course Unit Objective** object within the course variant – the objective will already have a type, such as Introduction, Skills Review, Learning Target, etc.)
* Assignment completion score
* Assignment mastery score
* Assessment completion score
* Assessment mastery score
* URL under which to find section-specific messaging for display in course pages

Relationships:

* 1 **Course Section Unit** contains N **Course Section Unit Objective** objects

### Catalog Course

Keyspace: **[main]**

Description: Each **Catalog Course** object represents a course listed in the University catalog. If the course corresponds to a **Course** object, it should have the same course ID. There will be many more courses in this table than the system manages. These are used mainly for the “Math Plan” service, and for analytics. These can come from scaping the catalog web site, or manual data entry.

Example: Every course in the catalog should be included, to support the tracking of students’ progress toward degree in a degree program.

Attributes:

* Course ID
* Title
* Minimum number of credits
* Maximum number of credits
* Prerequisite string (no attempt is made to parse this)
* Description

### Catalog Program

Keyspace: **[main]**

Description: Each **Catalog Program** object represents a degree program (or concentration) listed in the University catalog.

Example: Every undergraduate program in the catalog should be included, to support the tracking of students’ progress toward degree in a degree program.

Attributes:

* Program ID
* Title
* Description

TODO: The program requirements in terms of courses by term

## Application Queries

Note that the operations of “insert”, “delete”, and “update” are assumed for all tables, and won’t be listed here.

### Q1.2.1.1 Query all Term objects

### Q1.2.1.2 Query a single Term object by its term ID

### Q1.2.1.3 Query a single Term object by its active index (0 for active term, -1 for prior, +1 for next, etc.)

### Q1.2.1.4 Query all future Term objects (where active index is 1 or greater)

### Q1.2.2.1 Query for all Term Week objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.3.1 Query for all Pace Track objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.4.1 Query for all Pacing Structure objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.4.2 Query for a single Pacing Structure object by term ID and pacing structure ID (term ID selects keyspace)

### Q1.2.5.1 Query for all Rule Set objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.5.2 Query for a single Rule Set object by term ID and rule set ID (term ID selects keyspace)

### Q1.2.6.1 Query for all Rule Set Rule objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.6.2 Query for all Rule Set Rule objects in a term specified by term ID and rule set ID (term ID selects keyspace)

### Q1.2.6.3 Query for a single Rule Set Rule object in a term specified by term ID and rule set ID, with specified activity type and requirement (term ID selects keyspace)

### Q1.2.7.1 Query all Course objects

### Q1.2.7.2 Query a single Course object by its course ID

### Q1.2.8.1 Query all Course Prerequisite objects

### Q1.2.8.2 Query all Course Prerequisite objects for a specified course ID

### Q1.2.9.1 Query all Course Variant objects

### Q1.2.9.2 Query a single Course Variant object by its course ID and variant ID

### Q1.2.10.1 Query all Course Unit objects

### Q1.2.10.2 Query all Course Unit objects with a specified course ID and variant ID

### Q1.2.10.3 Query a single Course Unit object by its course ID, variant ID, and unit number

### Q1.2.11.1 Query all Course Unit Objective objects

### Q1.2.11.2 Query all Course Unit Objective objects with a specified course ID, variant ID, and unit number

### Q1.2.11.3 Query a single Course Unit Objective objects with a specified course ID, variant ID, unit number, and objective number

### Q1.2.12.1 Query all Lesson objects

### Q1.2.12.2 Query a single Lesson object with a specified lesson ID

### Q1.2.13.1 Query all Lesson Component objects

### Q1.2.13.2 Query all Lesson Component objects with a specified lesson ID

### Q1.2.14.1 Query all Course Unit Standards objects

### Q1.2.14.2 Query all Course Unit Standards objects with a specified course ID, variant ID, and unit number

### Q1.2.15.1 Query all Course Section objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.15.2 Query a single Course Section objects with a specified term ID, course ID, variant ID, and section number

### Q1.2.16.1 Query all Course Section Unit objects in a term specified by term ID (term ID selects keyspace)

### Q1.2.16.2 Query all Course Section Unit objects with a specified term ID, course ID, variant ID, and section number

### Q1.2.16.3 Query a single Course Section Unit object with a specified term ID, course ID, variant ID, section number, and unit number

### Q1.2.17.1 Query all Catalog Course objects

### Q1.2.17.2 Query a single Catalog Course object with specified course ID

### Q1.2.18.1 Query all Catalog Program objects

### Q1.2.18.2 Query a single Catalog Program object with specified program ID

## Logical Data Model

### Logical table: main.terms

|  |  |
| --- | --- |
| **main.terms** | |
| term\_id | K |
| start\_date |  |
| end\_date |  |
| academic\_year |  |
| active\_index |  |
| drop\_deadline\_date |  |
| withdraw\_deadline\_date |  |

Queries: Q1.2.1.1: query all term objects  
Q.1.2.1.2: query a single term object by its term ID

### Logical table: main.terms\_by\_active\_index

|  |  |
| --- | --- |
| **main.terms\_by\_active\_index** | |
| term\_id |  |
| start\_date |  |
| end\_date |  |
| academic\_year |  |
| active\_index | K |
| drop\_deadline\_date |  |
| withdraw\_deadline\_date |  |

Queries: Q1.2.1.3: query a single term object with a specified term index (0 for the active term)  
Q1.2.1.4: query all term objects with term index greater than 0

### Logical table: [term].term\_weeks

|  |  |
| --- | --- |
| **[term].term\_weeks** | |
| week\_number | K |
| start\_date |  |
| end\_date |  |

Queries: Q1.2.2.1: query for all term week objects in a specified term

### Logical table: [term].pace\_tracks\_by\_pace\_and\_track

|  |  |
| --- | --- |
| **[term].pace\_tracks** | |
| pace | K |
| track | K |
| subterm |  |
| criteria |  |

Queries: Q1.2.3.1: query for all pace track objects in a specified term

### Logical table: [term].pacing\_structures\_by\_pacing\_structure\_id

|  |  |
| --- | --- |
| **[term].pacing\_structures** | |
| pacing\_structure\_id | K |
| title |  |
| max\_courses |  |
| max\_open\_courses |  |
| schedule\_source |  |
| allow\_incompletes |  |
| min\_units\_to\_earn\_incomplete |  |
| require\_pacing\_structure\_exam |  |

Queries: Q1.2.4.1: query for all pacing structure objects in a specified term  
Q1.2.4.2: query for a single pacing structure object in a specified term with specified pacing structure ID

### Logical table: [term].rule\_sets

|  |  |
| --- | --- |
| **[term].rule\_sets** | |
| rule\_set\_id | K |
| title |  |
| require\_unit\_exams |  |
| require\_rule\_set\_exam |  |

Queries: Q1.2.5.1: query for all rule sets in a specified term  
Q1.2.5.2: query for the single rule set in a specified term with a specified rule set ID

### Logical table: [term].rule\_set\_rules\_by\_rule\_set\_id

|  |  |
| --- | --- |
| **[term].rule\_set\_rules\_by\_rule\_set** | |
| rule\_set\_id | K |
| activity\_type | C↑ |
| requirement | C↑ |

Queries: Q1.2.6.1: query for all rule set rule objects in a specified term  
Q1.2.6.2: query for all rule sets rules in a specified term by rule set ID

### Logical table: [term].rule\_set\_rules

|  |  |
| --- | --- |
| **[term].rule\_set\_rules** | |
| rule\_set\_id | K |
| activity\_type | K |
| requirement | K |

ueries: Q1.2.6.3: query for a single rule set rule in a specified term with specified activity and requirement

### Logical table: main.courses

|  |  |
| --- | --- |
| **main.courses** | |
| course\_id | K |
| title |  |
| label |  |
| inline\_prefix |  |
| catalog\_url |  |
| number\_of\_credts |  |
| type |  |

Queries: Q1.2.7.1: query for all course objects  
Q1.2.7.2: query for single course object by its course ID

### Logical table: main.course\_prerequisites\_by\_course\_id

|  |  |
| --- | --- |
| **main.course\_prerequisites\_course** | |
| course\_id | K |
| [prerequisite\_course\_id] | C↑ |
| [minimum\_grade\_requirement] |  |

Queries: Q1.2.8.1: query for all course prerequisite objects  
Q1.2.8.2: query for all course prerequisite objects for a specified course ID

### Logical table: main.course\_variants

|  |  |
| --- | --- |
| **main.course\_variants** | |
| course\_id | K |
| variant\_id | K |
| number\_of\_units |  |
| calculator\_type\_allowed |  |
| required\_etext\_id |  |

Queries: Q1.2.9.1: query for all course variant objects  
Q1.2.9.2: query for the single course variant object with specified course ID and variant ID

### Logical table: main.course\_units\_by\_course\_id\_and\_variant\_id

|  |  |
| --- | --- |
| **main.course\_units\_by\_course\_variant** | |
| course\_id | K |
| variant\_id | K |
| unit | C↑ |
| title |  |
| type |  |

Queries: Q1.2.10.1: query for all course unit objects  
Q1.2.10.2: query for all course unit objects with specified course ID and variant ID

### Logical table: main.course\_units

|  |  |
| --- | --- |
| **main.course\_units** | |
| course\_id | K |
| variant\_id | K |
| unit | K |
| title |  |
| type |  |

Queries: Q1.2.10.3: query for the single course unit object with specified course ID, variant ID, and unit

### Logical table: main.course\_unit\_objectives\_by\_course\_id\_and\_variant\_id\_and\_unit

|  |  |
| --- | --- |
| **main.course\_unit\_objectives\_by\_course\_variant\_unit** | |
| course\_id | K |
| variant\_id | K |
| unit | K |
| objective | C↑ |
| lesson\_number |  |
| lesson\_id |  |

Queries: Q1.2.11.1: query for all course unit objective objects  
Q1.2.11.2: query for all course unit objective objects with specified course ID, variant ID, and unit

### Logical table: main.course\_unit\_objectives

|  |  |
| --- | --- |
| **main.course\_unit\_objectives** | |
| course\_id | K |
| variant\_id | K |
| unit | K |
| objective | K |
| lesson\_number |  |
| lesson\_id |  |

Queries: Q1.2.11.3: query for the single course unit objective object with specified course ID, variant ID, unit, and objective

### Logical table: main.lessons

|  |  |
| --- | --- |
| **main.lessons** | |
| lesson\_id | K |
| type |  |
| description |  |

Queries: Q1.2.12.1: query for all lesson objects  
Q1.2.12.2: query for a single lesson object with a specified lesson ID

### Logical table: main.lesson\_components\_by\_lesson\_id

|  |  |
| --- | --- |
| **main.lesson\_components\_by\_lesson** | |
| lesson\_id | K |
| sequence\_number | C↑ |
| type |  |
| xml\_data |  |

Queries: Q1.2.13.1: query for all lesson component objects  
Q1.2.13.2: query for all lesson component objects with a specified lesson ID

### Logical table: main.course\_unit\_standards\_by\_course\_id\_and\_variant\_id\_and\_unit

|  |  |
| --- | --- |
| **main.course\_unit\_standards\_by\_coure\_variant\_unit** | |
| coure\_id | K |
| variant\_id | K |
| unit | K |
| standard\_id | C↑ |
| label |  |

Queries: Q1.2.14.1: query for all course units standard objects  
Q1.2.14.2: query for all course unit standard objects with a specified course ID, variant ID, and unit number

### Logical table: [term].course\_sections

|  |  |
| --- | --- |
| **[term].course\_sections** | |
| coure\_id | K |
| variant\_id | K |
| section | K |
| crn |  |
| aries\_start\_date |  |
| aries\_end\_date |  |
| start\_date |  |
| exam\_delete\_date |  |
| last\_student\_add\_Date |  |
| instruction\_type |  |
| exam\_structure |  |
| pacing\_group\_id |  |
| pacing\_structure |  |
| rule\_set\_id |  |
| is\_topmatter\_shown\_in\_practice |  |
| grading\_system |  |
| minimum\_score\_for\_a |  |
| minimum\_score\_for\_b |  |
| minimum\_score\_for\_c |  |
| minimum\_score\_for\_d |  |
| counts\_toward\_maximum\_open |  |
| is\_course\_label\_shown |  |
| is\_score\_shown |  |
| is\_grading\_scale\_shown |  |
| is\_distance |  |
| is\_online |  |
| Is\_bogus |  |
| bulk\_grade\_submission |  |
| proctoring\_options |  |
| topmatter |  |

Queries: Q1.2.15.1: query for all course section objects with a specified term ID  
Q1.2.15.2: query for the single course section object with a specified term ID, course ID, variant ID, and section number

### Logical table: [term].course\_section\_units\_by\_term\_id\_and\_course\_id\_and\_variant\_id\_and\_section\_number

|  |  |
| --- | --- |
| **[term].course\_section\_units\_by\_course\_variant\_section** | |
| coure\_id | K |
| variant\_id | K |
| section | K |
| unit | C↑ |
| skills\_review\_max\_score |  |
| skills\_review\_mastery\_score |  |
| unit\_review\_max\_score |  |
| unit\_review\_mastery\_score |  |
| unit\_exam\_max\_score |  |
| unit\_exam\_mastery\_score |  |
| final\_exam\_max\_score |  |
| final\_exam\_mastery\_score |  |
| homework\_max\_score |  |
| homework\_moveon\_score |  |
| homework\_mastery\_score |  |
| unit\_exam\_total\_tries\_allowed |  |
| unit\_exam\_tries\_after\_passing\_review |  |
| show\_testing\_window |  |
| first\_test\_date |  |
| last\_test\_date |  |
| test\_period\_start\_time |  |
| test\_period\_end\_time |  |
| award\_points\_for\_ontime\_review\_exam |  |
| topmatter |  |
| lesson\_course\_id |  |

Queries: Q1.2.16.1: query for all course section unit objects with a specified term ID  
Q1.2.16.2: query for all course section unit objects with a specified term ID, course ID, variant ID, and section number

### Logical table: [term].course\_section\_units

|  |  |
| --- | --- |
| **[term].course\_section\_units** | |
| coure\_id | K |
| variant\_id | K |
| section\_number | K |
| unit | K |
| skills\_review\_max\_score |  |
| skills\_review\_mastery\_score |  |
| unit\_review\_max\_score |  |
| unit\_review\_mastery\_score |  |
| unit\_exam\_max\_score |  |
| unit\_exam\_mastery\_score |  |
| final\_exam\_max\_score |  |
| final\_exam\_mastery\_score |  |
| homework\_max\_score |  |
| homework\_moveon\_score |  |
| homework\_mastery\_score |  |
| unit\_exam\_total\_tries\_allowed |  |
| unit\_exam\_tries\_after\_passing\_review |  |
| show\_testing\_window |  |
| first\_test\_date |  |
| last\_test\_date |  |
| test\_period\_start\_time |  |
| test\_period\_end\_time |  |
| award\_points\_for\_ontime\_review\_exam |  |
| topmatter |  |
| lesson\_course\_id |  |

Queries: Q1.2.16.3: query for the single course section unit object with a specified term ID, course ID, variant ID, section number, and unit

### Logical table: [main].catalog\_courses

|  |  |
| --- | --- |
| **main.catalog\_courses** | |
| course\_id | K |
| title |  |
| minimum\_number\_of\_credits |  |
| maximum\_number\_of\_credits |  |
| prerequisite\_string |  |
| description |  |

Queries: Q1.2.17.1: query for all catalog course objects  
Q1.2.17.2: query for all catalog course objects with a specified course ID

### Logical table: [main].catalog\_programs

|  |  |
| --- | --- |
| **main.catalog\_programs** | |
| program\_id | K |
| title |  |
| description |  |

Queries: Q1.2.18.1: query for all catalog program objects  
Q1.2.18.2: query for all catalog program objects with a specified program ID

# Section 2. Student Service

### Conceptual Data Model

#### New Student

Student ID

Academic level

Registration type

Term

#### Student

Student ID

Internal ID

Last name

First name

Preferred name

Middle initial

Screen name

Application term

Class level

College

Department

Program

Minor

Anticipated graduation term

Number of transfer credits

High school CEEB code

High school class rank

High school class size

Hich school GPA

ACT math score

SAT math score

AP Calculus score

Residency

Birthdate

Gender

Has disciplinary history?

Disciplinary status

Hold severity

Timelimit factor

Number of extension days

Is licensed?

Campus

Email

Adviser email

Admit type

Course order enforced

Rule set ID

Admitted?

When created

Canvas ID

Is real student?

#### Student Accommodation

Student ID

Term

When created

Accommodations

Comment

Interviewer

#### Student Category

Student ID

Category

Start date

End date

#### Student Discipline

Student ID

Incident date

Incident type

Course ID

Unit

Cheating description

Action type

Action comment

Interviewer

Proctor

#### Student Hold

Student ID

Hold ID

Severity

When added

Number of times applied

Admin message

Student message

#### Student Visit

Student ID

When entered

When exited

Location

Seat

#### Live Advisee

Student ID

PIDM

First name

Last name

Adviser ID

Adviser PIDM

Adviser first name

Adviser last name

#### Live Student

Student ID

Internal ID

Last name

First name

Preferred name

Middle initial

College

Department

Program

Major

Minor

High school CEEB code

High school class rank

High school class size

Hich school GPA

ACT math score

SAT math score

SAT revised score

AP Calculus score

Residency

Admission Term

Admit type

Estimated graduation date

Birthdate

Gender

Email

Adviser email

State

Campus

# Section 3. Profile Service

### Conceptual Data Model

#### Profile

Profile ID

Title

##### 1 HAS N [Profile Question]

#### Profile Question

Term

Question number

Description

Question type

Course ID

Mandatory?

Question HTML

##### 1 HAS N [Profile Question Choice]

#### Profile Question Choice

Choice number

Choice

Description

Choice meaning

Choice HTML

#### Student Profile Response

Student ID

Profile ID

Question number

When submitted

Answer

# Section 4. Registration Service

### Conceptual Data Model

#### Student Course Registration

Term

Student ID

Course ID

Section

Pace order

Open status

Grading option

Is complete?

Course score

Course grade

Course grade GPA

Prerequisite satisfied?

On initial class roll?

On final class roll?

Placed by exam type

Zero unit

Instruction type

Registration status

Last class roll date

Is provisional?

Is synthetic?

##### 1 HAS N [Student Course Unit]

##### 1 HAS N [Student Course Mastery]

#### Student Course Unit

Term

Student ID

Course ID

Unit

Review Exam Status (n/a, not attempted, not passed, passed on time, passed late)

Review exam score

Review exam points

Proctored Exam Status (n/a, not attempted, not passed, passed on time, passed late)

Proctored exam score

Proctored exam points

##### 1 HAS N [Student Lesson]

#### Student Lesson

Objective

Lesson ID

Seed

Last component finished

When lecture viewed

#### Student Course Mastery

Score

Number of standards mastered in first half

Number of standards mastered in second half

Number of standards eligible for mastery

##### 1 HAS N [Student Unit Mastery]

##### 1 HAS N [Student Exploration Mastery]

#### Student Unit Mastery

Unit

Score

Skills review status (“Passed” or number of questions correct so far)

##### 1 HAS N [Student Unit Standard Mastery]

#### Student Unit Standard Mastery

Standard status (Eligible, attempted, mastered on time, mastered late)

Number of times question 1 has been answered correctly

Number of times question 2 has been answered correctly

#### Student Exploration Mastery

Exploration number

#### Duplicate Registration

Term

Student ID

Course ID

Section

Pace order

Open status

Grading option

Is complete?

Course score

Course grade

Course grade GPA

Prerequisite satisfied?

On initial class roll?

On final class roll?

Placed by exam type

Zero unit

Instruction type

Registration status

Last class roll date

#### Guest Student

Student ID

Course ID

Section

Unit

Course enrolled

Section enrolled

Required to submit homework?

#### Student Incomplete

Term

Student ID

Course ID

Section Number

Incomplete in progress?

Incomplete counts toward pace?

Incomplete term

Incomplete deadline date

#### Student Transfer Credit

Student ID

Course ID

Number of credits

Grade

Grade GPA

When added

When credit refused

#### Required Prerequisite

Course ID

List of prerequisite course IDs

May be concurrent?

List of Prereqiuisite grades

#### Live CSU Credit

Student ID

Term

Course ID

Course number

Credits

Grade

Grade GPA

#### Live Registration

Term

Anticipated graduation term

Student ID

Internal ID

Last name

First name

Class level

College

Department

Major

Number of transfer credits

High school CEEB code

High school class rank

High school class size

Hich school GPA

ACT math score

SAT math score

AP Calculus score

Residency

Birthdate

Gender

Email

Adviser email

Campus

Admit type

Course ID

Section number

Grading option

Registration status

Instruction type (Advance placement, Credit by exam, resident instruction, continuing education, continuous registration, unknown)

#### Live Transfer Credit

Student ID

Term

Transferred course number

Course ID

Credits

Grade

Grade GPA

#### Cohort

Cohort ID

Size

Instructor

# Section 5. Resource Service

### Conceptual Data Model

#### Resource

Resource ID

Type

Description

Days allowed

Holds allowed

Hold ID

#### Testing Calculator Loan

Student ID

Calculator number

Exam serial number

When issued

When returned

#### Student Resource Loan

Student ID

Resource ID

When loaned

When due

When returned

Number of times overdue message displayed

# Section 6. Schedule Service

### Conceptual Data Model

#### Calendar

Date

Date type (drop deadline, start date 1, start date 2, end date 1, end date 2, start date 1 next, end date 1 next, holiday, bookstore deadline, tutorial start, tutorial end, tutorial info, walk-in placement session, placement maintenance start N, placement maintenance end N, summer last-add date)

##### 1 HAS N [Calendar Open Time]

#### Calendar Open Time

Opening time

Closing time

Weekdays

#### Standard Milestone

Pace track

Pace

Pace Index

Unit

Objective

Milestone type (Opening date, mastery due date, exploration due date, exploration 1-day late date, course deadline)

Milestone date

##### 1 HAS N [Student Standard Milestone]

#### Student Standard Milestone

Student ID

Milestone date

#### Milestone

Term

Pace

Pace track

Number

Course index

Unit

Type (Skills Review, unit review, unit exam, final exam, final exam last try)

Date

Attempts allowed

##### 1 HAS N [Student Milestone]

#### Student Milestone

Student ID

Milestone date

Attempts allowed

#### Student Pace Summary

Student ID

Pace

Pace track

Course

Section

Pace order

Incomplete in progress?

Milestone number

Milestone unit

Milestone date

New milestone date

Exam date

Review exam points

#### Student Term

Term

Student ID

Pace

Pace track

First course ID

Cohort

Urgency

Do not disturb?

#### Student Appeal

Student ID

Term

When created

Relief given?

Pace

Pace track

Milestone number

Milestone date

New deadline date

Number of attempts allowed

Circumstances

Comment

Interviewer

# Section 7. Placement Service

### Conceptual Data Model

#### Challenge Fee

Student ID

Course ID

Exam date

Billed date

#### Placement Exam

Exam ID

Maximum number of online attempts

Maximum number of proctored attempts

#### Placement Fee

Student ID

Course ID

Exam date

Billed date

#### Placement Log

Serial number

Student ID

Course ID

Exam ID

When started

When finished

When recovered

Academic year

#### Placement Score Queue

Internal ID

Test code

Test date

Test score

#### Unproctopred Placement

Term

Application Term

Course ID

Start date

End date

#### Student Placement Attempt

Term

Serial number

Student ID

Exam ID

Revision

Instance

When started

When finished

Exam label

Course ID

Sequence number

Academic year

Last name

First name

Middle initial

Score

Passed?

How validated

##### 1 HAS N [Student Placement Attempt Answer]

##### 1 HAS N [Student Placement Attempt Subtest Score]

#### Student Placement Attempt Answer

When finished

Question number

Answer

Correct?

Subtest

Tree reference

#### Student Placement Attempt Subtest Score

Subtest name

Score

#### Student Placement Credit

Student ID

Course ID

Placement result

Serial number

Exam ID

Exam source

When finished

When credit refused

#### Student Placement Denied

Student ID

Course ID

Placement result

Serial number

Exam ID

Exam source

When finished

Why denied

#### Student Challenge Attempt

Term

Serial number

Student ID

Exam ID

Revision

Instance

When started

When finished

Exam label

Course ID

Sequence number

Academic year

Last name

First name

Middle initial

Score

Passed

How validated

##### 1 HAS N [Student Challenge Attempt Answer]

#### Student Challenge Attempt Answer

When finished

Question number

Answer

Correct?

#### Student Math Plan

Session tag

Student ID

Internal ID

Survey ID

Page ID

Question number

When submitted

Answer

# Section 8. Assessment Service

### Conceptual Data Model

#### Assignment

Assignment ID

Type (homework | standard | exploration)

Course ID

Unit

Objective

Tree reference

Title

When activated

When pulled

#### Mastery Exam

Exam ID

Type (standard)

Course ID

Unit

Objective

Tree reference

Title

Button label

When activated

When pulled

#### Exam

Exam ID

Type (Skills review, unit review, unit, final, licensing, placement, challenge)

Course ID

Unit

Tree reference

Title

Button label

When activated

When pulled

#### Item

Item ID

(todo)

#### Standard

Standard ID

Mastery groups (bit field)

##### 1 HAS N [Standard Item]

#### Standard Item

Item ID

Mastery group number

Practice group number

Formative?

Placement?

#### Student Exam

Term

Serial number

Student ID

Exam ID

Revision

Instance

When started

When finished

Exam label

Course ID

Sequence number

Exam score

Time OK?

Passed? (yes or no)

Is first passing attempt?

Unit

Exam type (Skills review, unit review, unit, final, licensing, placement, challenge)

Exam source

##### 1 HAS N [Student Exam Answer]

#### Student Exam Answer

Question number

Answer index

Objective

Answer

Student ID

Exam ID

Correct?

When finished

Subtest

#### Student Homework

Serial number

Student ID

Homework ID

Revision

Instance

When started

When finished

Homework label

Course ID

Section number

Unit

Objective

Type

Score

Time OK?

Passed?

##### 1 HAS N [Student Homework Answer]

#### Student Homework Answer

Question number

Answer index

Objective

Answer

Student ID

Correct?

When finished

#### Student Licensing Attempt

Term

Student ID

Exam ID

When started

When finished

Serial number

Exam score

Passed?

Calculus course ID

Exam label

#### Student Mastery Attempt

Serial number

Exam ID

Student ID

When started

When finished

Exam score

Mastery score

Passed (yes, no, ignored, taken away)

Is first passing attempt?

Exam source (remote, testing center, hand-graded)

#### Student Mastery Attempt Answer

Serial number

Exam ID

Question number

Is correct? (yes or no)

#### Student Lesson Assignment

Student ID

Course

Lesson ID

When shown

When open

When hidden

When started

When finished

Score in tenths

#### Student Standard

##### Student ID

##### Standard ID

##### When placed out

##### When mastered

##### Mastery groups

##### Completed steps

# Section 9. E-Text Service

### Conceptual Data Model

#### E-Text

Etext ID

Retention (yes, if completed, no)

Purchase URL

Refund period

Allow manyal key entry?

Active?

Description

##### 1 HAS N [E-text Course]

##### 1 HAS N [E-Text Key]

#### E-Text Course

Course ID

#### E-Text Key

Key

When activated

#### Student E-Text

Student ID

Etest ID

Etext key

When active

Expiration date

Refund deadline date

When refunded

Refund reason

# Section 10: Messaging Service

### Conceptual Data Model

#### Message

Term

Touch point

Message code

Subject

Template

#### Message Lookup

#### Student Message

Student ID

When sent

Pace

Course index

Touch point

Message code

Sender

# Section 11. Testing Center Service

### Conceptual Data Model

#### Testing Center

Testing center ID

Name

When created

When approved

Is active?

Is remote?

Is proctored?

##### 1 HAS N [Testing Station]

#### Testing Station

Computer ID

Station number

Description

Icon X

Icon Y

Usage

State

When created

When approved

Current student ID

Current course ID

Current unit

Current exam ID

#### Pending Exam

Serial number

Exam ID

Student ID

When started

Start time, in minutes

Sequence number

Course ID

Unit

Exam type (Skills Review, etc.)

Time limit factor

Role ID

# Section 12 . Admininstrative Service

### Conceptual Data Model

#### Database Info

Description

#### High School

CEEB code

Name

Address

City

State

Zip code

#### Hold Type

Hold ID

Severity (fatal, non-fatal)

Can staff add?

Can staff delete?

Description

Student message

#### Local Login

Username

Student ID

Default role ID

Password salt

Password hash

Stored key

Server key

When created

When expires

When last logged in

Force password change?

Email

Password fails allowed

Password fails so far

Disabled?

Why disabled

#### Parameters

Program name

List of String parameters

Date parameter

#### User Permission

Username

Function

Permission level

#### Tree Path

ID

Parent ID

Depth

Sort Order

Label

#### Zip Code

Zip code

City

State

# Section 13. Analytics Service

### Conceptual Data Model

#### Analytics Student

Student ID

Internal ID

Last name

First name

Birthdate

GPA

Sex

Hispanic/Latinx race?

Maer/Ind race?

Asian race?

Black race?

Hawaiian race?

White race?

Multiple races?

Application term

Admission term

Admit program

High school CEEB code

High school graduation date

High school GPA

High school class rank

High school class size

ACT math

SAT math

SAT revised math