A Curriculum Design System For Information Systems Programs

James M. Slack
slack@mnsu.edu
Computer Information Science
Minnesota State University
Mankato, MN 56001, USA

Abstract

This paper describes how we used a Curriculum Design System to redesign our Information Systems program. It also discusses the benefits of using such a system, which include automated generation of curriculum documentation and automated validation of the curriculum.

Keywords: curriculum design, automated curriculum validation, IS 2002, model curriculum

1. INTRODUCTION

A few years ago, our existing Computer and Information Sciences (CIS) Department split into two departments: Computer Science and Information Systems & Technology. (It has since been renamed to Computer Information Science.) The university charged our new department with offering two new majors, one of which was Information Systems (IS). We were given the task of designing a new IS curriculum.

Fortunately, the old CIS department offered a Management Information Systems (MIS) major, so we were not starting from scratch. Nevertheless, we decided to begin with a clean slate, because the existing MIS major had grown in a haphazard way over many years, and was definitely showing its age. Furthermore, we realized that we had a unique opportunity to develop a high-quality program.

We used the IS 2002 Model Curriculum (Gorgone, Davis, Valacich, Topi, Feinstein, & Longenecker, 2003) to guide our development. Several authors, such as (Dwyer & Knapp, 2004; Waguespack Jr, 2004), have explained how they used IS 2002 to improve their IS programs. In contrast, this paper describes how we used IS 2002 to create a new Curriculum Design System (CDS), and how we use the CDS to aid our IS curriculum design.

2. REQUIREMENTS

The first requirement of any curriculum design system is that it should store course information, including course number, title, description, and other obvious attributes. It should also ensure that there are no redundant prerequisites and that the prerequisite structure does not have cycles. We also wanted our CDS to generate a *roadmap*: a four-year, semester-by-semester course schedule that satisfies all prerequisites while keeping the number of credits per semester within a small range. This roadmap shows how students can complete the entire program in four years.

Our university requires that each program (major, minor, etc.) have a set of goals, each of which must derive from one or more of seven university-wide goals. Our college has a similar requirement with eighteen goals. The same university or college goal can derive several program goals. Another requirement for the CDS then, was that it should map program goals to institutional goals and vice versa.

A CDS should store *knowledge units*, each of which is a small group of related topics with a learning goal and a set of measurable objectives to assess whether students have met that goal. Knowledge units are associated with courses in a many-to-many relationship: each course covers many knowledge units, and several courses may cover the same knowledge unit.

Usually, when several courses cover the same knowledge unit, they cover it at different levels. For example, a freshman *Introduction to IS* course may require students to define a database management system as a set of programs for managing related data, whereas a senior capstone course might expect students to develop a database and use it in a program. Therefore, for each knowledge unit covered in a course, the CDS should keep track of the *knowledge level* of that knowledge unit along with each of its associated objectives.

We wanted the CDS to validate the data as much as possible, to avoid curriculum-related errors we had encountered in the past, such as duplicate or missing prerequisites, a roadmap with nonexistent courses, or a program that students could not complete in four years.

Another requirement for the CDS is that it should support assessment of courses in the program and the program itself. For each knowledge unit taught in a course, the CDS should let us specify how to assess whether the objectives of that knowledge are satisfied.

A related requirement is that the CDS support accreditation activities, namely, recording assessment data in each course, periodic evaluations of that assessment data, and records of continual improvement. This requirement was not a high priority because it is tangential to the primary purpose of the CDS: designing a new IS curriculum. However, course assessment is a natural extension of the system.

An ambitious requirement, which we were not sure we could accomplish, was that the CDS should produce publication-quality documents. The primary document we had in mind, of course, was the program proposal, but we also envisioned other possibilities, such as a course catalog and syllabi. Automatic document production has several benefits; the main one is accuracy. Course information is incredibly complex, especially when combined with knowledge units and objectives. Documents that the CDS generates are more likely to be free of errors than those produced manually.

We made a decision early in our curriculum design process that the new IS program would have no electives. The rationale was simplification of counseling, student scheduling, and course scheduling. Fortunately, having no electives also simplified design of the CDS considerably, although this was not a factor in the decision.

3. DATA MODEL DESIGN

Figure 1 shows a simplified version of the CDS data model. (Figure 5 in the appendix has a more thorough data model.)

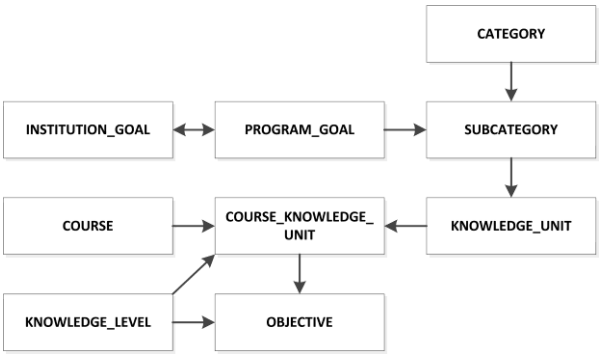


Figure 1: Data model (simplified).

Institution goals relate to program goals in a many-to-many relationship. We organized knowledge units into categories and subcategories, and associated each subcategory with a single program goal. This design allows tracking of institutional or program goals throughout the system.

The COURSE_KNOWLEDGE_UNIT table forms a many-to-many relationship between courses and knowledge units. Each knowledge unit used in a course has an associated knowledge level. Each objective also has an associated knowledge level.

4. IMPLEMENTATION

In contrast to the rather careful design of the data model, our implementation of the CDS was decidedly *ad hoc*; it consisted of many SQL queries and Python scripts written in a hurry. This informal development style was necessary largely because we were under enormous time pressure, but also because we did not know

exactly what we would need beforehand. Fortunately, we teach information systems, after all (!), so we were comfortable running scripts and working directly with the database contents.

Two of the most challenging CDS requirements were validation of the curriculum data and the ability to generate documents. We wrote Python scripts to satisfy both of these requirements.

Our Python validation script saved us from many potentially embarrassing omissions and oversights. Some checks this script performed include:

- The roadmap must schedule prerequisites before the requiring course.
- Each course in the roadmap must appear in the semester during which the university normally offers it.
- All courses must appear in the roadmap.
- Every course covers enough, but not too many knowledge units. (The minimum and maximum values are configurable.)
- Every course/knowledge unit combination has objectives.
- The knowledge level of each objective is compatible with (i.e., less than or equal to) its associated course/knowledge unit combination.
- Each course/knowledge unit combination has at least one objective with the same knowledge level as the knowledge level of the course/knowledge unit combination.
- The description of each objective contains a keyword that matches the knowledge level of that objective, and does not contain any keywords from higher knowledge levels.
- The knowledge levels do not have any keywords in common with each other.

Normal database constraints (e.g., primary keys, foreign keys, and uniqueness constraints) also helped to keep the data consistent.

Although the document generation capability was a major undertaking, it really paid off. Figure 2 shows the automated process the CDS used. A database table stored documents. Each such document consisted of a Mako (Mako, 2010) template, which provides looping, selection, and the ability to insert database content. Each document used reStructuredText (DocUtils, 2010) markup syntax, which allows generation of HTML or LaTeX output. Once in LaTeX format, the pdflatex utility generates Portable Document Format (PDF) documents.

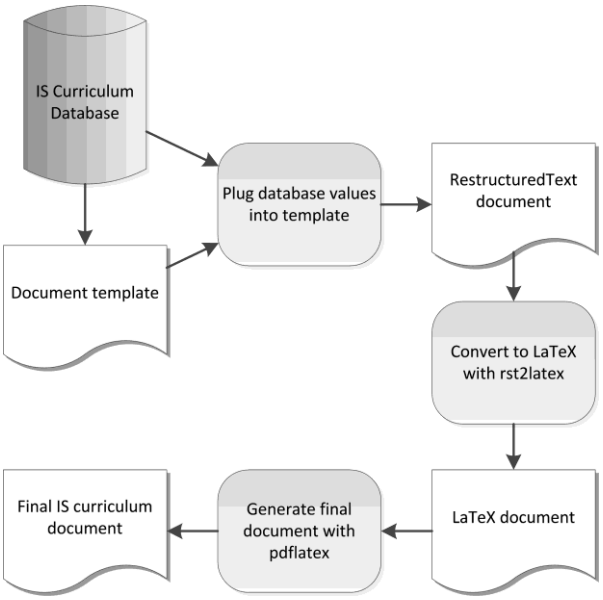
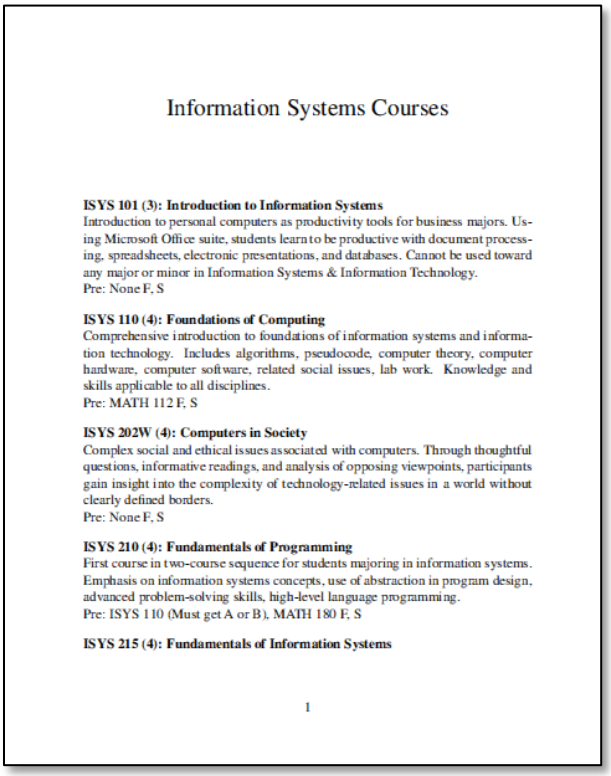


Figure 2: Document generation process.

```
Information Systems Courses
===========================
<%
courses = (db.query(Course)
    .filter(Course.designator == 'ISYS')
    .order_by(Course.designator, Course.number))
%>

% for c in courses:
**${c.designator} ${c.number}${c.extra}
(${c.credits}):
${c.name}**

${c.description}

Pre: ${c.prerequisites} ${c.schedule}

% endfor
```

Figure 3 shows the first page of the document the CDS generates from this template.

The previous example generates a PDF document that mimics our official university course catalog for the IS program. It shows that we used an Object/Relational Mapping (ORM) tool, SQLAlchemy (SQLAlchemy, 2010), to simplify working with the database.



Information Systems Courses

ISYS 101 (3): Introduction to Information Systems
Introduction to personal computers as productivity tools for business majors. Using Microsoft Office suite, students learn to be productive with document processing, spreadsheets, electronic presentations, and databases. Cannot be used toward any major or minor in Information Systems & Information Technology.
Pre: None F, S

ISYS 110 (4): Foundations of Computing
Comprehensive introduction to foundations of information systems and information technology. Includes algorithms, pseudocode, computer theory, computer hardware, computer software, related social issues, lab work. Knowledge and skills applicable to all disciplines.
Pre: MATH 112 F, S

ISYS 202W (4): Computers in Society
Complex social and ethical issues associated with computers. Through thoughtful questions, informative readings, and analysis of opposing viewpoints, participants gain insight into the complexity of technology-related issues in a world without clearly defined borders.
Pre: None F, S

ISYS 210 (4): Fundamentals of Programming
First course in two-course sequence for students majoring in information systems. Emphasis on information systems concepts, use of abstraction in program design, advanced problem-solving skills, high-level language programming.
Pre: ISYS 110 (Must get A or B), MATH 180 F, S

ISYS 215 (4): Fundamentals of Information Systems

1

Figure 3: Generated course catalog document.

One seemingly difficult feature that turned out to be trivial was the ability to generate a nicely formatted prerequisite diagram. The Graphviz package (Graphviz, 2010) made this job easy. Figure 4 shows an example.

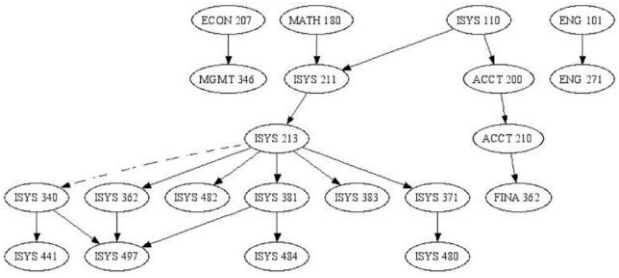


Figure 4: Generated prerequisite diagram.

5. USING THE CDS

We started using the CDS by entering institutional goals into the system. We then drew on studies of current IS topics, such as those in (Noll & Wilkins, 2002; Janicki, Kline, Gowan, & Konopaske, 2004) to develop goals for our new IS program:

1. Each student will gain a sound foundation in computing basics: analysis and design, programming, testing, software development, security, database, and human-computer interaction.

2. Each student will assimilate a solid base of business enterprise concepts, including principles of accounting, finance, business law, management, operations, and enterprise resource planning (ERP).

3. Each student will learn the theory and practice of information technology, and develop skills to apply this knowledge to analyze and solve business problems.

4. Each student will develop analytical, critical thinking, and interpersonal skills applicable to real-world problems.

5. Each student will develop effective oral and written communication skills.

6. Each student will appreciate the social and ethical issues in information systems.

We then associated each of these goals with one or more institutional goals.

Because we wanted an accreditable program, we started with knowledge units from the IS 2002 Model Curriculum. This curriculum organizes its knowledge units into a two-level hierarchy of categories and subcategories. Following advice in (Daigle, Longenecker Jr, Landry, & Pardue, 2004), we modified the categories, subcategories, and knowledge units until we came up with a set that worked for our needs.

For knowledge levels, we used Bloom's Taxonomy (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). In increasing order, the levels are:

1. *Knowledge*: Recall or recognize information. For example, "List at least four primitive types in Java." Identified by keywords such as *define*, *list*, and *recall*.

2. *Comprehension*: Understand the meaning of information. For example, "Explain the difference between a primitive type and an object type in Java." Identified by keywords such as *classify*, *distinguish*, and *explain*.

3. *Application*: Use the information in a new way. For example, "Design a database consisting of at least four interrelated tables." Identified by keywords such as *apply*, *compute*, *construct*, and *implement*.

4. *Analysis*: Break down a concept into component parts to understand its underlying structure. For example, "Analyze the following tasks to determine requirements." Identified by keywords such as *analyze*, *compare*, *contrast*, and *differentiate*.

5. *Synthesis*: Use learned concepts to create something new. For example, "Write a Java program that looks for SQL injection flaws in a web application." Identified by keywords such as *assemble*, *categorize*, *develop*, *organize*, and *summarize*.

6. *Evaluation*: Assess the value of concepts. For example, "Critique the quality of this Java program with respect to robustness, security, and ease of maintenance." Identified by keywords such as *assess*, *critique*, *evaluate*, and *prove*.

We spent a couple of months developing courses that would cover as many knowledge units as possible within the credit limit imposed by the university. We started with a blend of existing courses from our old MIS program and the recommended courses in the IS 2002 model curriculum, and iterated until we were comfortable with the result.

The next step was to develop objectives. During this step, the CDS validation script saved us a lot of time by catching countless errors. For example, it made sure each objective contained an appropriate keyword for the desired knowledge level. Although this check did not prevent *all* un-measurable objectives, it did provide a safety net. The validation script also ensured that objectives were compatible with their associated course/knowledge unit combinations.

Table 1 shows the size of each table in the CDS.

Database Table	Rows
ASSESSMENT_AREA	8
CATEGORY	9
COURSE	32
COURSE_KNOWLEDGE_UNIT	185
COURSE_TYPE	5
DOCUMENT	3
GOAL_LEVEL	3
INSTITUTION_GOAL	25
INSTITUTION_TO_PROGRAM_GOAL	16
KNOWLEDGE_LEVEL	6
KNOWLEDGE_UNIT	136
OBJECTIVE	1,033
PREREQUISITE	22
PROGRAM_GOAL	6
SEMESTER	10
SUBCATEGORY	30

Table 1: Number of rows of each database table.

The most complex document the CDS produced was the curriculum proposal, which was more than 100 pages long. In spite of the length, however, the document template was only about 400 lines. The CDS generated the curriculum proposal from data stored in the database. The resulting proposal consisted of the following sections:

- An introduction.
- A list of courses by type (core, general education, support, prerequisites, and others).
- A list of courses and their prerequisites, along with a diagram showing the prerequisite structure.
- A course roadmap, showing courses to take during each of eight semesters that satisfies all prerequisites and includes all courses.

- A list of all knowledge units and their descriptions, grouped by categories and subcategories.

- A list of all courses in the program (our department along with supporting departments). For each course, the catalog description, schedule, prerequisites, course type, and knowledge units. For each knowledge unit, the category, (program) goal, measurable objectives with knowledge level. (See Figure 6 in the appendix for an example.)

- Knowledge unit coverage, showing the courses that cover each knowledge unit, and at which knowledge level. (See Figure 7 in the appendix for an example.)

The CDS also produced many other documents for submitting the program proposal, creating web site content, and advising students, including:

- A mockup of the official university course catalog (shown earlier).

- A list of knowledge units for each course.

- Sample syllabi for each course, including instructor information (stored in the database), catalog description, topics along with number of hours spent for each topic and knowledge level (from knowledge unit information), objectives, textbooks, and boilerplate.

- The semester-by-semester roadmap of courses to take in a four-year program.

- Knowledge units and objectives across all courses.

- Comparison of knowledge units (to look for duplication).

- Objectives for each program goal, by course.

- Objectives for each program goal, by knowledge unit.

- Various HTML documents for the departmental web site.

As we continued to work on the curriculum, we started to appreciate the time we saved by not having to update all these documents manually every time we made a change. Just as unit tests make programmers more willing to improve their code without worrying about stupid mistakes, we felt free to make many tweaks to our IS curriculum right up to the proposal submission deadline. In all, we generated countless versions, and printed about ten complete proposal versions by the deadline date.

6. THE FUTURE

Although our CDS has served us well, the fact that it consists of a database along with *ad hoc* SQL and Python scripts makes it unwieldy to use on a day-to-day basis. Our highest priority for the CDS is to consolidate its functionality in a web-based application. This change alone would make the system much more accessible and usable.

The system stores assessment information for each course and knowledge unit, but does support regular assessment activities. In particular, we would like to use it in a future accreditation effort, and the CDS is nowhere near ready for that. We would like to add that support.

The CDS has unit tests for its validation feature, but it needs more test cases for the remainder of the system.

Besides IS, our department also offers an Information Technology (IT) major. The IT major has electives, which the current CDS system does not support. We would like to extend the CDS system to support electives, so we can use it for both of our majors.

Beyond that, we would also like to generalize the system so other departments in our university and in other institutions can use it. Making the system easy to use for non-computer-related disciplines is probably the most challenging task we have set for ourselves.

For our own program, we plan to examine how the new IS 2010 Model Curriculum (Topi, et al., 2010) might improve our curriculum, and then incorporate and resulting changes in the CDS.

7. CONCLUSIONS

Out of sheer necessity, we have developed a custom curriculum design system that applies to our specific needs. In spite of the substantial effort in the design and implementation of this system, the enormous benefits of automated validation and automated generation of curriculum documents have really paid off. We are very satisfied with our resulting IS program.

We hope to improve the CDS system over time and make a new web-enabled version of it available to others.

8. REFERENCES

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives: the classification of educational goals; Handbook I: Cognitive Domain.* New York: Longmans, Green.

Daigle, R., Longenecker Jr, H., Landry, J., & Pardue, J. (2004). Using the IS 2002 model curriculum for mapping an IS curriculum. *Information Systems Education Journal , 2* (1), 3-6.

DocUtils. (2010). Retrieved from reStructuredText: Markup Syntax and Parser Component of Docutils: http://docutils.sourceforge.net/rst.html

Dwyer, C., & Knapp, C. (2004). How Useful is IS 2002? A Case Study Applying the Model Curriculum. *Journal of Information Systems Education , 15* (4), 4090416.

Gorgone, J., Davis, G., Valacich, J., Topi, H., Feinstein, D., & Longenecker, H. (2003). IS 2002 model curriculum and guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems , 11* (1), 1.

Graphviz. (2010). Retrieved from Graphviz Graph Visualization Software: http://www.graphviz.org/

Janicki, T., Kline, D., Gowan, J., & Konopaske, R. (2004). Matching employer needs with IS curriculum: An exploratory study. *Information Systems Education Journal , 2* (21), 3.

Mako. (2010). Retrieved from Mako Templates for Python: http://www.makotemplates.org/

Noll, C., & Wilkins, M. (2002). Critical skills of IS professionals: A model for curriculum development. *Journal of Information Technology Education , 1* (3), 143-154.

SQLAlchemy. (2010). Retrieved from SQLAlchemy: The Python SQL Toolkit and Object Relational Mapper: http://www.sqlalchemy.org/

Topi, H., Valacich, J., Wright, R., Kaiser, K., Nunamaker Jr, J., Sipior, J., et al. (2010). IS 2010: curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems , 26* (1), 18.

Waguespack Jr, L. (2004). An Implementation of IS2002: BSCIS'04-Bentley College. *The Proceedings of the Information Systems Education Conference 2004.*

Appendices

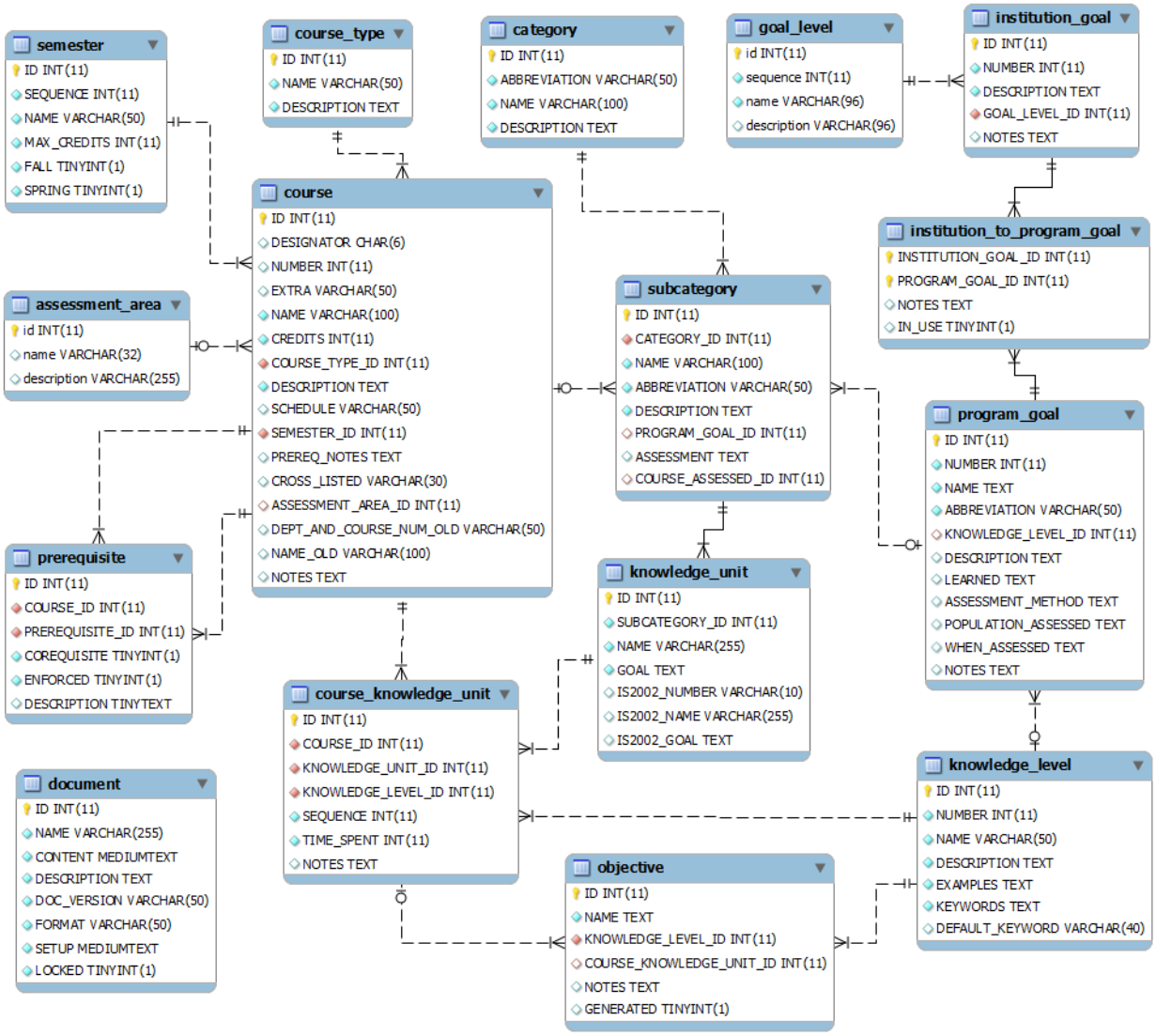
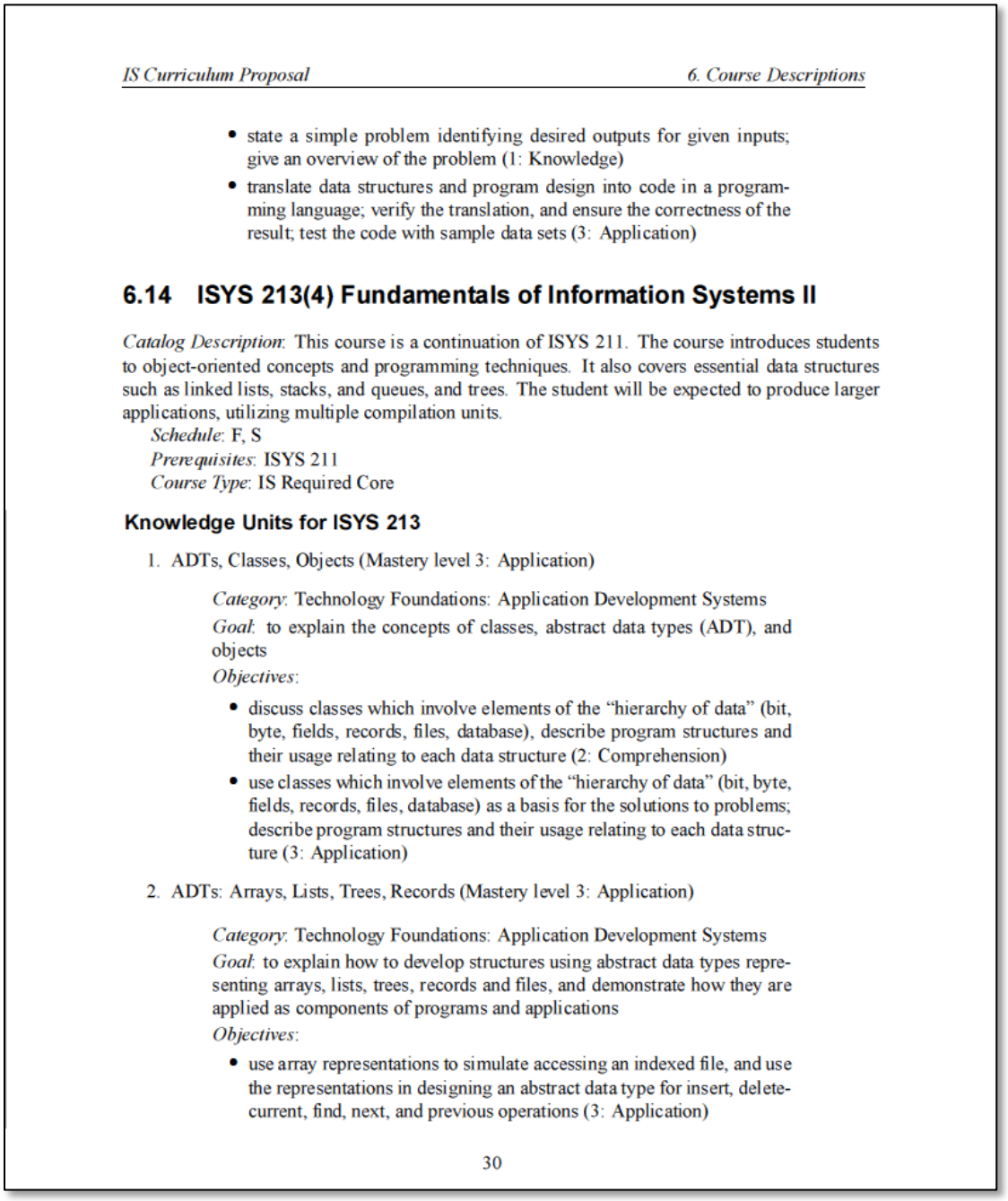


Figure 5: Partial data model for the curriculum design system.



IS Curriculum Proposal *6. Course Descriptions*

- state a simple problem identifying desired outputs for given inputs; give an overview of the problem (1: Knowledge)
- translate data structures and program design into code in a programming language; verify the translation, and ensure the correctness of the result; test the code with sample data sets (3: Application)

6.14 ISYS 213(4) Fundamentals of Information Systems II

Catalog Description: This course is a continuation of ISYS 211. The course introduces students to object-oriented concepts and programming techniques. It also covers essential data structures such as linked lists, stacks, and queues, and trees. The student will be expected to produce larger applications, utilizing multiple compilation units.

Schedule: F, S
Prerequisites: ISYS 211
Course Type: IS Required Core

Knowledge Units for ISYS 213

1. ADTs, Classes, Objects (Mastery level 3: Application)

   Category: Technology Foundations: Application Development Systems
   Goal: to explain the concepts of classes, abstract data types (ADT), and objects
   Objectives:

   - discuss classes which involve elements of the "hierarchy of data" (bit, byte, fields, records, files, database), describe program structures and their usage relating to each data structure (2: Comprehension)
   - use classes which involve elements of the "hierarchy of data" (bit, byte, fields, records, files, database) as a basis for the solutions to problems; describe program structures and their usage relating to each data structure (3: Application)

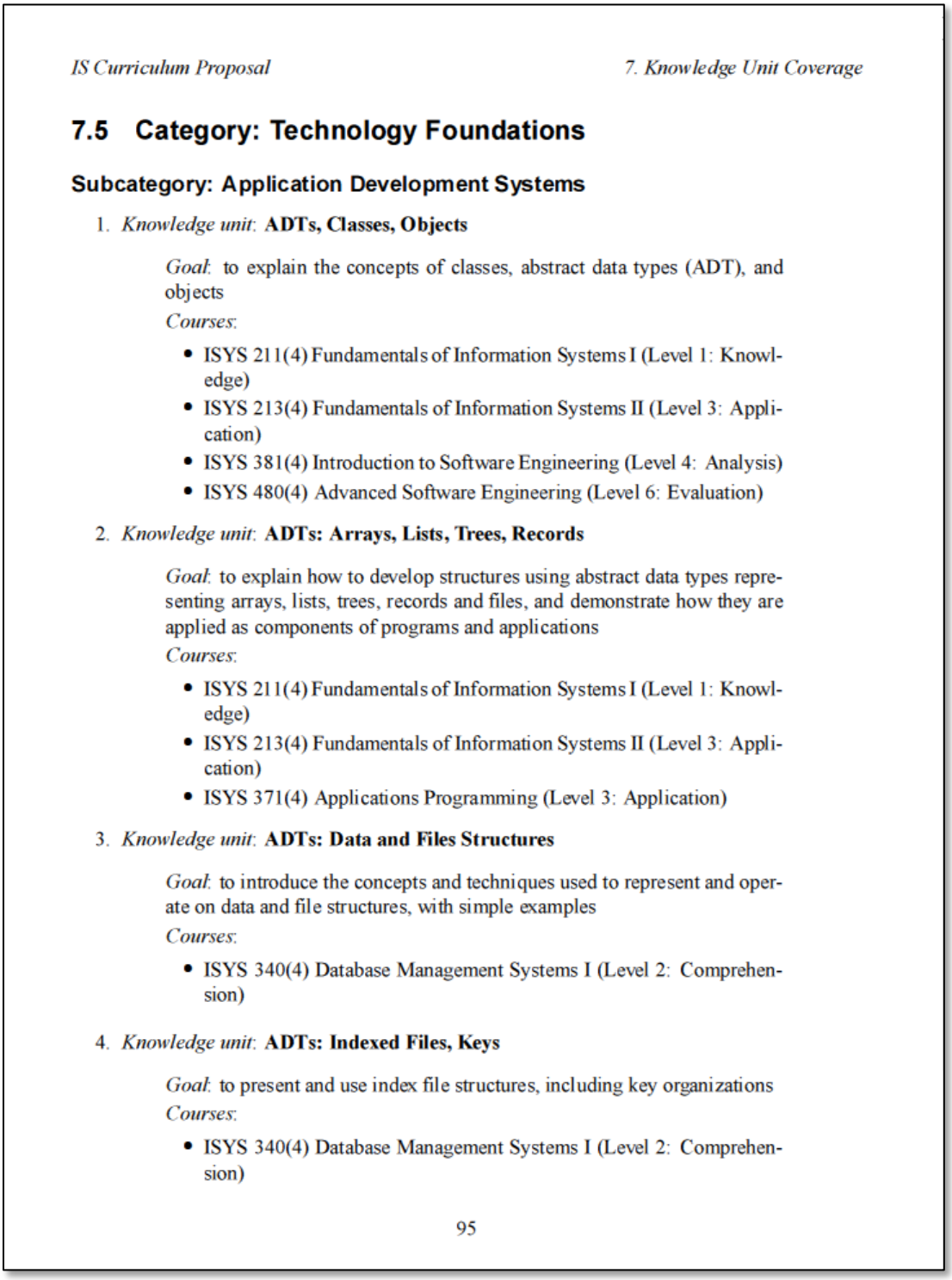
2. ADTs: Arrays, Lists, Trees, Records (Mastery level 3: Application)

   Category: Technology Foundations: Application Development Systems
   Goal: to explain how to develop structures using abstract data types representing arrays, lists, trees, records and files, and demonstrate how they are applied as components of programs and applications
   Objectives:

   - use array representations to simulate accessing an indexed file, and use the representations in designing an abstract data type for insert, delete-current, find, next, and previous operations (3: Application)

30

Figure 6: An example page from the IS Program Proposal showing a generated course description.



IS Curriculum Proposal *7. Knowledge Unit Coverage*

7.5 Category: Technology Foundations

Subcategory: Application Development Systems

1. *Knowledge unit*: **ADTs, Classes, Objects**

   Goal: to explain the concepts of classes, abstract data types (ADT), and objects

   Courses:

   - ISYS 211(4) Fundamentals of Information Systems I (Level 1: Knowledge)
   - ISYS 213(4) Fundamentals of Information Systems II (Level 3: Application)
   - ISYS 381(4) Introduction to Software Engineering (Level 4: Analysis)
   - ISYS 480(4) Advanced Software Engineering (Level 6: Evaluation)

2. *Knowledge unit*: **ADTs: Arrays, Lists, Trees, Records**

   Goal: to explain how to develop structures using abstract data types representing arrays, lists, trees, records and files, and demonstrate how they are applied as components of programs and applications

   Courses:

   - ISYS 211(4) Fundamentals of Information Systems I (Level 1: Knowledge)
   - ISYS 213(4) Fundamentals of Information Systems II (Level 3: Application)
   - ISYS 371(4) Applications Programming (Level 3: Application)

3. *Knowledge unit*: **ADTs: Data and Files Structures**

   Goal: to introduce the concepts and techniques used to represent and operate on data and file structures, with simple examples

   Courses:

   - ISYS 340(4) Database Management Systems I (Level 2: Comprehension)

4. *Knowledge unit*: **ADTs: Indexed Files, Keys**

   Goal: to present and use index file structures, including key organizations

   Courses:

   - ISYS 340(4) Database Management Systems I (Level 2: Comprehension)

95

Figure 7: An example page from the IS Program Proposal showing generated knowledge-unit coverage.

Data Dictionary

Table: ASSESSMENT_AREA

Major assessment areas, such as ethical/social, critical thinking.

Column	Type	Size	Comment
ID	int		primary key
NAME	varchar	32	name of the assessment area
DESCRIPTION	varchar	255	description of the assessment area

Table: CATEGORY

Knowledge unit major categories.

Column	Type	Size	Comment
ID	int		primary key
ABBREVIATION	varchar	50	short version of category name
NAME	varchar	100	"official" name of category
DESCRIPTION	text	65535	description of category

Table: COURSE

Information about individual courses in the curriculum.

Column	Type	Size	Comment
ID	int		primary key
DESIGNATOR	char	6	department or program designator, such as ISYS
NUMBER	int		course number, such as 100
EXTRA	varchar	50	extra course characters, such as W for writing intensive
NAME	varchar	100	name of the course
CREDITS	int		number of credits
COURSE_TYPE_ID	int		type of the course (core, required, etc.)
DESCRIPTION	text	65535	catalog description of the course
SCHEDULE	varchar	50	when the course is normally offered, e.g., F, S
SEMESTER_ID	int		which semester a student should take this course (roadmap)
PREREQ_NOTES	text	65535	additional notes about prerequisites, e.g., A or B required
CROSS_LISTED	varchar	30	list of courses cross listed with this one
ASSESSMENT_AREA_ID	int		assessment area for this course
DEPT_AND_COURSE_NUM_OLD	varchar	50	previous course designator and number
NAME_OLD	varchar	100	previous course name
NOTES	text	65535	private notes about the course

Table: COURSE_KNOWLEDGE_UNIT

Units of knowledge covered in a course.

Column	Type	Size	Comment
ID	int		primary key
COURSE_ID	int		course for this knowledge unit / course pair
KNOWLEDGE_UNIT_ID	int		knowledge unit for this knowledge unit / course pair
KNOWLEDGE_LEVEL_ID	int		knowledge level (bloom's taxonomy)
SEQUENCE	int		order this should appear in reports
TIME_SPENT	int		relative time spent on this knowledge unit in this course
NOTES	text	65535	private notes

Table: COURSE_TYPE

Types of course in the curriculum, e.g., Required, Core, etc.

Column	Type	Size	Comment
ID	int		primary key
NAME	varchar	50	name of the course type
DESCRIPTION	text	65535	description of the course type

Table: DOCUMENT

Document templates that database values can be plugged into to produce a report.

Column	Type	Size	Comment
ID	int		primary key
NAME	varchar	255	name of the document
CONTENT	mediumtext	16777215	content of the document (usually a template)
DESCRIPTION	text	65535	description of what the document produces
DOC_VERSION	varchar	50	version of the document
FORMAT	varchar	50	format of the document (e.g., html, markdown)
SETUP	mediumtext	16777215	setup that should display before the document (e.g., title page)
LOCKED	tinyint		prevents accidental modification of the content

Table: GOAL_LEVEL

Levels of institutional goal, e.g., university, college, etc.

Column	Type	Size	Comment
ID	int		primary key
SEQUENCE	int		order the goal levels should appear
NAME	varchar	96	name of the goal level, e.g., university, college
DESCRIPTION	varchar	96	description of the goal level

Table: INSTITUTION_GOAL

Overarching goals of the institution.

Column	Type	Size	Comment
ID	int		primary key
NUMBER	int		number of the goal (for sequencing)
DESCRIPTION	text	65535	description of the goal
GOAL_LEVEL_ID	int		level of the goal (e.g., university, college, ...)
NOTES	text	65535	private notes

Table: INSTITUTION_TO_PROGRAM_GOAL

Association between institutional and program goals.

Column	Type	Size	Comment
INSTITUTION_GOAL_ID	int		associated institution goal
PROGRAM_GOAL_ID	int		associated program goal
NOTES	text	65535	private notes
IN_USE	tinyint		whether this association is in use

Table: KNOWLEDGE_LEVEL

Mastery levels, e.g., knowledge, understanding, application, etc.

Column	Type	Size	Comment
ID	int		primary key
NUMBER	int		number of the level, e.g., 1=knowledge, 2=understanding
NAME	varchar	50	name of the level, e.g., knowledge, understanding
DESCRIPTION	text	65535	description of the level
EXAMPLES	text	65535	examples for display in a GUI
KEYWORDS	text	65535	unique list of keywords that apply to this level
DEFAULT_KEYWORD	varchar	40	default keyword for generated knowledge units

Table: KNOWLEDGE_UNIT

Units of knowledge that a student should master.

Column	Type	Size	Comment
ID	int		primary key
SUBCATEGORY_ID	int		subcategory for this knowledge unit
NAME	varchar	255	name of this knowledge unit
GOAL	text	65535	learning goal of this knowledge unit
IS2002_NUMBER	varchar	10	IS2002 knowledge unit number
IS2002_NAME	varchar	255	IS2002 knowledge unit name
IS2002_GOAL	text	65535	IS2002 knowledge unit content

Table: OBJECTIVE

Measureable objectives that shows student mastery of a knowledge unit in a course.

Column	Type	Size	Comment
ID	int		primary key
NAME	text	65535	name of this objective
KNOWLEDGE_LEVEL_ID	int		knowledge level for this objective, e.g., knowledge, understanding, etc.
COURSE_KNOWLEDGE_UNIT_ID	int		course and knowledge unit this objective applies to
NOTES	text	65535	private notes
GENERATED	tinyint		whether this objective was generated automatically

Table: PREREQUISITE

Course prerequisites.

Column	Type	Size	Comment
ID	int		primary key
COURSE_ID	int		course that has the prerequisite
PREREQUISITE_ID	int		prerequisite for the course
COREQUISITE	tinyint		whether this prerequisite can be taken with the course
ENFORCED	tinyint		whether this prerequisite is enforced
DESCRIPTION	tinytext	255	description of this prerequisite

Table: PROGRAM_GOAL

Goals of the specific program.

Column	Type	Size	Comment
ID	int		primary key
NUMBER	int		number of the goal for sequencing
NAME	text	65535	full name of the program goal
ABBREVIATION	varchar	50	abbreviation for this program goal
KNOWLEDGE_LEVEL_ID	int		knowledge level of this goal (knowledge, understanding, etc.)
DESCRIPTION	text	65535	description of the goal
LEARNED	text	65535	what is learned in achieving this goal
ASSESSMENT_METHOD	text	65535	explanation of how this goal is assessed
POPULATION_ASSESSED	text	65535	population that is assessed
WHEN_ASSESSED	text	65535	when the goal is assessed
NOTES	text	65535	private notes

Table: SEMESTER

Semesters in a 4-year roadmap, e.g., Junior Fall.

Column	Type	Size	Comment
ID	int		primary key
SEQUENCE	int		order the semesters should appear
NAME	varchar	50	name of the semester, e.g. Junior Fall
MAX_CREDITS	int		maximum credits a student should take during this semester
FALL	tinyint		true if this is a fall semester
SPRING	tinyint		true if this is a spring semester

Table: SUBCATEGORY

Knowledge unit minor categories.

Column	Type	Size	Comment
ID	int		primary key
CATEGORY_ID	int		category to which this subcategory belongs
NAME	varchar	100	name of this subcategory
ABBREVIATION	varchar	50	abbreviation for this subcategory
DESCRIPTION	text	65535	description of this subcategory
PROGRAM_GOAL_ID	int		program goal this subcategory achieves
ASSESSMENT	text	65535	how this subcategory is assessed