ITN 256: Database Design Project

Michael J. Smith

ITN 256

Oct. 1, 2012

Updated: Nov.12, 2012

CONTENTS

Section 1: Project Plan

Introduction

For this project, I chose to design a database that will help me keep track of my college program. The reason I chose this particular option is because I find that this will be the most practical application of a database for me, and therefore the most useful to me.

Disadvantages of Using File Processing System (Excel)

Up to this point, I have been maintaining and tracking this information with a series of Excel spreadsheets, which requires entering the same information into multiple spreadsheets. Sometimes, I get careless and forget to update the information across all the spreadsheets, which affects the accuracy of the data.

Spreadsheets also limit the ability to easily manipulate the data. The spreadsheets I currently use to track my college program don't allow me to easily track which certificates I can earn using courses I have already taken. They also don't show me which certificates I am close to earning based on courses I have already taken. Without this ability, I could end up one class short (or a few classes short) of fulfilling the requirement for a certificate.

Advantages of Using Database

Designing and implementing a database will be advantageous in this instance because it will allow me to more accurately and easily maintain and track my degree progress.

Using a database will eliminate the need for multiple spreadsheets, which will mean that I only need to update the data in one place, increasing the consistency of the data.

I will also be able to manipulate the data easier with a database. I will be able to track my fulfillment of the requirements for certificates and plan future courses accordingly to ensure that I do fulfill the requirements, thus maximizing the benefit from the money spent taking courses.

Project Description

The project is to design and implement a database to track my progression through my NVCC degree in the Information Systems Technology (IST) program.

The database will help keep track of the following data:

- **Degree progress** – This will allow me to see what courses I need to complete my degree, as well as other potential degrees that I may only need one or two courses (on top of the courses I am taking for my current degree) to complete.
- **Certificate qualifications** – This will allow me to see what certificates I can receive with the courses required for my degree, and what certificates I may need an extra course or two (on top of the courses I am taking for my current degree) to complete.
- **Grades** – This will allow me to track my grades by semester, by course type (online or in-person), and my cumulative degree. I could also track my grades by instructor and other attributes.
    - NOTE: The reason I am not tracking my program GPA – only for IST – is because I don't need to take general education courses due to my completion of a previous degree. So, all my courses are program-specific.
- **Instructor and classmate contacts** – This will allow me to expand my professional network, which increases my chances of landing better jobs or, if needed, securing a recommendation for future jobs.

BENEFITS OF DATABASE

- DATA UPDATED REGULARLY
  The data is constantly being updated, which is time consuming and inefficient.
- GENERATE REPORTS
  Reports can be used to check how different course decisions affect my degree progress.
- SIMPLIFICATION
  Data only needs to be updated once, improving accuracy of data and efficiency.

Reports

With this database, reports can be generated to track my degree progress, as well as show how different course decisions affect my degree progress and graduation date. Reports can also show which certificates I qualify for based on the courses I've taken, or what courses I need to

take to complete the certificate requirement. I will also be able to track my GPA by semester and my cumulative GPA. Lastly, I will also be able to maintain a list of contacts including both professors and students, that will help me expand my professional network.

Sample report that can be generated from the database:

Table 1. Sample Report

THIRD SEMESTER (SUMMER 2012)

COURSE ID	COURSE TITLE	CREDITS	COST	PREREQUISITES	GRADE	% GRADE
ITE 221	PC Hardware and OS Architecture	3	$413.25	ITE 115	A	89.7%
ITD 110	Web Design I	3	$413.25	ITE 115	A	98.9%
ITN 200	Administration of Network Resources	3	$413.25	ITN 101	A	97.3%
ITN 260	Network Security Basics	3	$413.25	ITE 115	A	99.5%
TOTAL		12	$1,653.00	**AVERAGES**	A	96.4%

Entities

My database will contain the following entities with the following attributes:

- Course
  - **Attributes:** Course Name, Course Grade, Credits, Description, Cost, Instructor
- Degree
  - **Attributes:** Required Courses (Course), Credits, Elective Credits
- Certificates
  - **Attributes:** Courses Taken (Course), Credits, Courses Needed

These may change but this is the initial framework from which I am working.

Metadata

My database will contain the following metadata:

Table 2. Metadata

DATA ITEM						
NAME	TYPE	LENGTH	MIN	MAX	DESCRIPTION	SOURCE
Number	Alphanumeric	30			Course ID number	NVCC Registrar
Title	Alphanumeric	30			Name of course	NVCC Registrar
Credits	Integer	1	2	4	Number of credits	NVCC Registrar
Cost	Decimal	7			Cost of course	NVCC Registrar
Prerequisite	Alphanumeric	6			Course prerequisite	NVCC Registrar
Grade	Alphanumeric	1			Grade received in class	Academic Unit
% Grade	Decimal	4	00.0	100.00	Percent grade received.	Academic Unit

Database Application

For my database, I propose using a personal database. This category of database application is best for my project because it will allow me to save time and work efficiently. As I mentioned before, I spend a lot of time entering the same information across several different spreadsheets, which is inefficient and opens my data up to errors and inaccuracies.

Using a personal database will allow me to store all my college program information in an efficient manner.

Another reason that a personal database application will work best for my project is because this database contains data of interest only to me. There is little reason to share the database information with anyone else, so I think that this is the best application for me. Also, because the database is personal in nature, there is no reason to believe that it will ever grow to a point where it needs to be shared with anyone else. Thus, a personal database will work best.

Section 2: Logical Design

Introduction

As I mentioned before, my database will help me keep track of my college degree program. Listed below are the detailed entities for my database. This list expands upon and is updated from the preliminary list on found in **Entities** list of this document.

Table 2-1. Expanded Entity List

ENTITY TYPE	ATTRIBUTE
Course	Course ID
	Course Number
	Course Title
	Course Cost
	Course Prerequisite
	Instructor ID
Grade	Grade ID
	Course ID
	Instructor ID
	Course LTR Grade
	Course PCT Grade
Instructor	Instructor ID
	Course ID
	Instructor First Name
	Instructor Last Name
	Instructor Email
	Instructor Phone
Classmate	Classmate ID
	Classmate First Name
	Classmate Last Name
	Classmate Email
	Classmate Phone
Shared Course	Course ID
	Classmate ID
Certificate Assignment	Course ID
	Certificate ID
	Course Taken
	Course Needed
Certificate	Certificate ID
	Certificate Name
Credit	Course ID
	Credit Value
Semester	Course ID
	Semester Date

The DEGREE entity that was mentioned in Section 1 has been scrapped because, after considering it further, I will only qualify one degree, which means that I do not need to keep track of multiple degrees for the database.

Entity Relationship Diagram

The simple entity relationship diagram (ERD) for this database can be found in Figure 2-1. I have omitted the attributes at this point, to simplify the diagram.

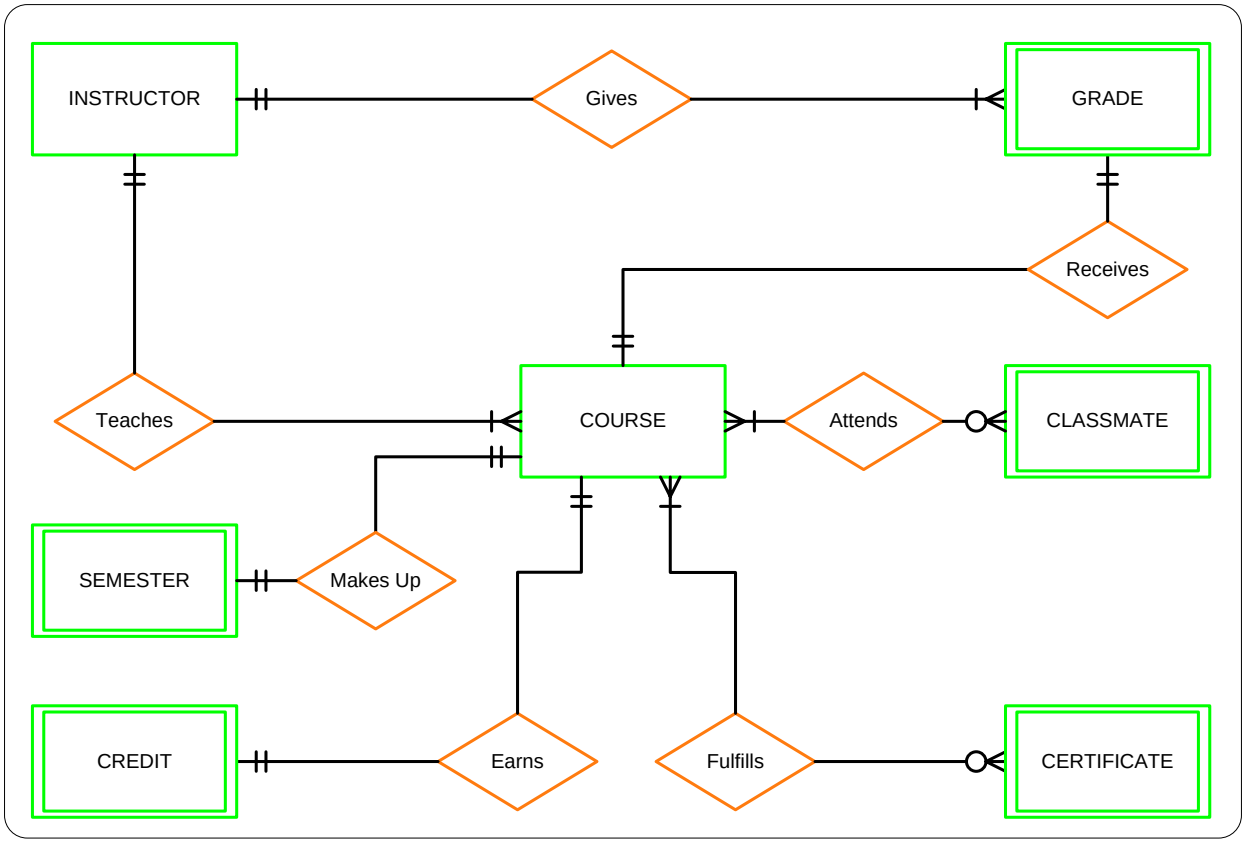

Figure 2-1. Simple Entity Relationship Diagram

Table 2-2. Entity Definitions

INSTRUCTOR	An instructor is person who teaches a course, and gives a grade.
GRADE	A grade is an instructor's evaluation of performance in a course.
COURSE	A course is a class that is taught by an instructor, may fulfill a certificate requirement, makes up a semester, is given a grade, earns credit, and may be attended by a classmate.
CLASSMATE	A classmate is a member of the same class in college; attends a course.
SEMESTER	A semester is a specific term of an academic year.
CREDIT	A credit is a unit awarded when a sufficient grade is received in a course; signals completion of the course.
CERTIFICATE	A certificate is a document that certifies that a student has successfully completed the required courses to earn the certificate.

Relationship Definitions

The "Teaches" relationship links an instructor to a course they are teaching, and a course to the instructor who is teaching it. An instructor must teach at least one course, but can teach many courses. Each course must have one and only one instructor teaching it.

The "Gives" relationship links an instructor to the grade the instructor gives for performance in a class. An instructor must give at least one grade, but can give many grades, depending on the number of courses he or she teaches. A grade must be given by one and only one instructor.

The "Receives" relationship links a grade, given by an instructor, to a course. A course must receive one and only one grade. A grade must be given to only one course. GRADE is a weak entity because it depends on the relationship between the instructor and the course.

The "Attends" relationship links a classmate to the courses we both attend. Because this database is personal in nature (only applies to me), the only data I want to keep track of in regards to classmates is if we have a class together. Once that requirement is met, there is no need for the database to track all of the courses we have together. So, this is a many-to-many relationship because a classmate must attend one course with me but can attend many. It has a mandatory cardinality because a classmate must share at least one course with me. A course may have zero classmates attend it or it may have many classmates attend it. This relationship has an optional cardinality. CLASSMATE is a weak entity because a person is a classmate only if they attend a course with me. Without the course entity, this entity would not exist.

The "Fulfills" relationship links a course to the certificate requirement it satisfies. A course can satisfy zero, one, or many certificate requirements. But, a certificate requirement must be fulfilled by at least one course. The COURSE-to-CERTIFICATE relationship has an optional many cardinality, while the CERTIFICATE-to-COURSE relationship has a mandatory many cardinality. This is a weak entity because it would not exist without the COURSE entity.

The "Makes Up" relationship links a course to a semester. A course must be linked to one and only one semester. Similarly, a semester must be linked to one and only one course. SEMESTER is a weak entity because without the COURSE entity, it would not exist.

The "Earns" relationship links a course to the credit I receive for completing it. A course must earn one and only one credit value. Likewise, a credit value must be earned by only one course. CREDIT is a weak entity because without the COURSE entity, it wouldn't exist.

Business Rules

I. An instructor must teach each course.

The first business rule that I have for my database is that an instructor must teach every course. This relationship is a one-to-many relationship because an instructor can teach any number of courses. But, each course must have one instructor assigned to it. This relationship has a mandatory cardinality.

II. An instructor must give a grade to each course.

My second business rule states that an instructor must give each course a grade in order for it to be considered complete. This relationship is also one-to-many because one instructor can give any number of grades, depending on how many courses the instructor teaches. It has a mandatory cardinality.

III. A course must receive a grade.

This business rule states that every course must receive a grade. This relationship is a one-to-one with a mandatory cardinality because one course must receive one grade. That is, a course must have a grade associated with it.

IV. A classmate must attend one course.

A classmate must attend one course with me. For the scope of this project, once a classmate fulfills this requirement, they are included in the database. I have no need to keep track of all of the courses we have in common. Thus, this is a one-to-many relationship instead of a many-to-many relationship. It has a mandatory cardinality. A course, however, does not have to be attended by a classmate. This means it has an optional cardinality.

V. A course may meet the requirements of any number of certificates.

This business rule states that each course may meet the requirements of any number of certificates. Or, a course does not have to meet any certificate requirements. This relationship is a many-to-many relationship with an optional cardinality.

VI. A course must earn credit.

Each course must earn credit towards a degree because courses that do not earn credit are, ultimately for me, a waste of money. This relationship is a one-to-one relationship. This relationship has a mandatory cardinality.

VII. A course must be part of a semester.

Each course must be part of a semester. This relationship is a one-to-one relationship because a course must be associated with one semester date and a semester date can have only one course associated with it. This relationship has a mandatory cardinality.

Associative Entities

Because the relationship between COURSE and CERTIFICATE is a many-to-many relationship, it requires an associative entity in order to be resolved. The associative entity is shown in Figure 2-2.

Figure 2-2. Certificate Associative Entity

The CERTIFICATE ASSIGNMENT associative entity resolves the many-to-many relationship between COURSE and CERTIFICATE. It associates a specific course with a specific certificate. Each course can fulfill more than one certificate requirement (or none). And, each certificate requirement can be fulfilled by more than one course. It has a composite key made up of Course ID and Certificate ID.

The relationship between CLASSMATE and COURSE is also a many-to-many relationship requiring an associative entity. The entity is shown in Figure 2-3.

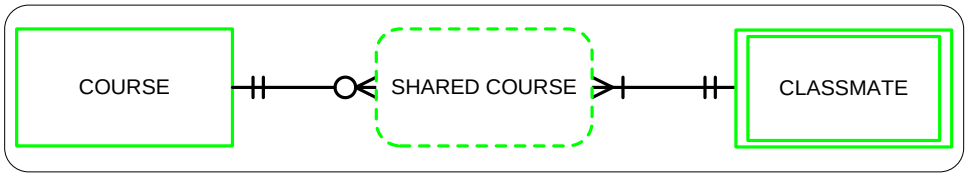

Figure 2-3. Classmate Associative Entity

The SHARED COURSE associative entity resolves the many-to-many relationship between CLASSMATE and COURSE by associating a specific classmate with a specific course. It has a composite key made up of Course ID and Classmate ID.

Constraints

Entity Integrity

This database will use entity integrity, meaning all the primary keys in the database will have valid data values. None of them will be allowed to be null.

Referential Integrity

This database will also use referential integrity. Each foreign key will match a primary key value in another relation, per the definition given in the textbook, *Modern Database Management*. (Hoffer, Ramesh, and Topi, 2011).

This database contains the following tables: COURSE, GRADE, INSTRUCTOR, CLASSMATE, SHAREDCOURSE, CERTIFICATE, CERTIFICATEASSIGNMENT, CREDIT, and SEMESTER. Each course in the COURSE table must be associated with an instructor in the INSTRUCTOR table, a grade in the GRADE table, a credit value in the CREDIT table, and a semester value in the SEMESTER table.

While a course doesn't have to be associated with a CERTIFICATE or a CLASSMATE, both of those must be associated with a COURSE in order to exist in the database.

Figure 2-4 shows referential integrity being reinforced.

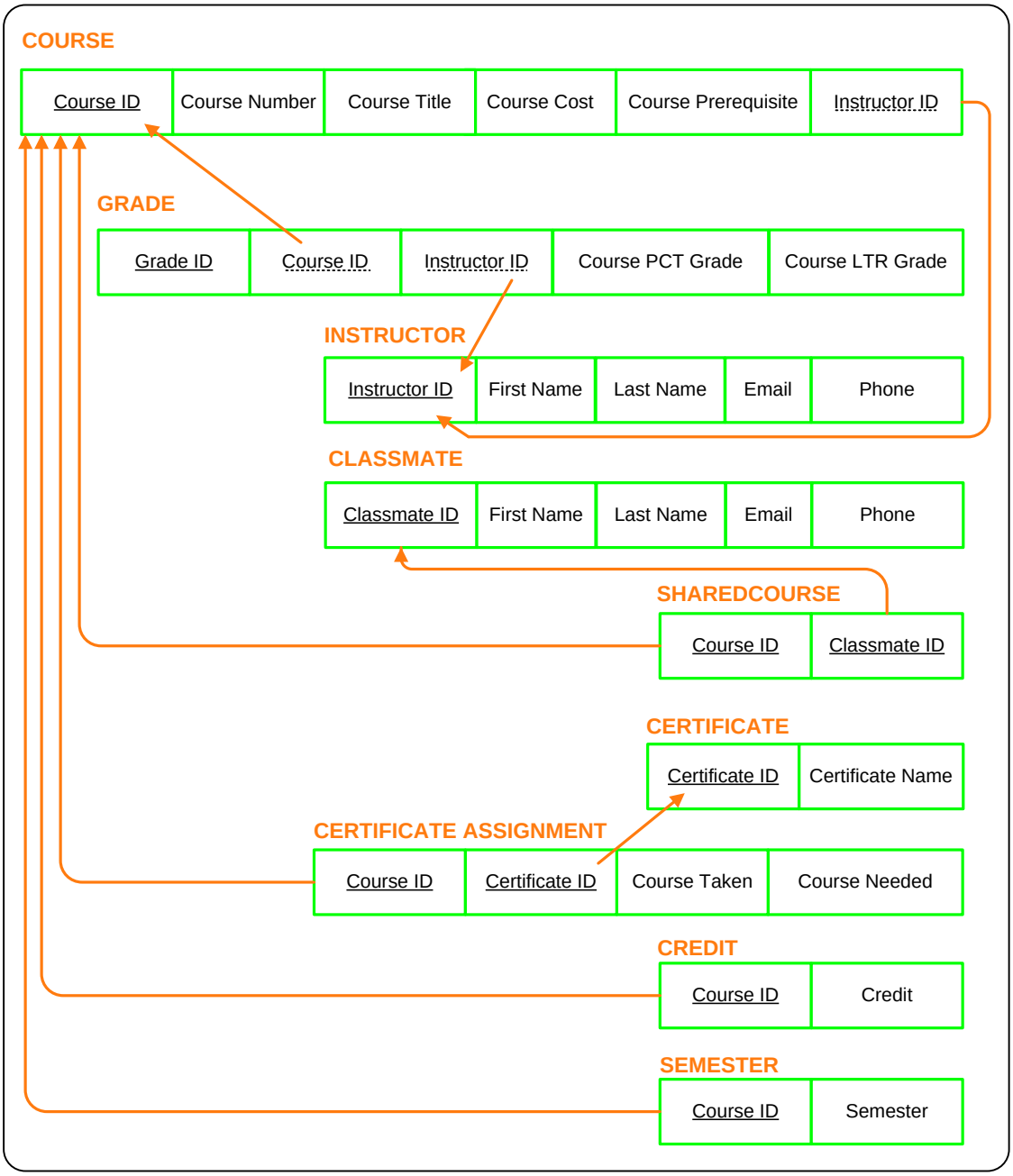

Figure 2-4. Referential Integrity Being Reinforced

Normalization

Following the steps outlined in the text (Hoffer, Ramesh, and Topi, 2011, Pgs. 182-186), my database needs to be normalized. As part of step 0, I have created a user view, which consists of a report showing course information including, number, title, cost, and instructor information.

Table 2-3. Sample User View

COURSE ID	COURSE NUMBER	COURSE TITLE	COURSE COST	INSTRUCTOR ID	FIRST NAME	LAST NAME
2	ITN 100	Intro To Telecom	$413.25	4	Hatim	Hussein
3	ITE 115	Intro To Computer Apps & Concepts	$413.25	5	Hugo	Serrato
4	ITE 130	Intro To Internet Services	$413.25	6	Laura	Albert
5	ITE 170	Multimedia Software	$413.25	9	Wanda	Mally
6	ITP 100	Software Design	$413.25	8	Richard	Eichers
7	ITN 171	Unix I	$413.25	7	Sami	Saleh
8	ITN 101	Network Concepts	$413.25	10	Charles	Kellermann
9	ITD 110	Web Design I	$413.25	11	Peggy	McKelvey
10	ITN 260	Network Security Basics	$413.25	14	Zenaida	Bodwin
11	ITE 221	PC Hardware & OS Architecture	$413.25	12	Shawn	Bailey
12	ITN 200	Admin of Network Resources	$413.25	13	Nima	Zahadat
13	ITN 208	Protocols & Communication	$581.00	13	Nima	Zahadat

The sample user view above shows the table in first normal form because the repeating groups have been eliminated, and the primary key (Course ID) has been defined.

Functional Dependencies

Based on the user view above, the following functional dependencies have been identified.

Table 2-4. Functional Dependencies

COURSE ID →	Course Number, Course Title, Course Cost, Instructor ID, First Name, and Last Name
INSTRUCTOR ID →	First Name and Last Name

There are no partial dependencies in this table because each row is uniquely identified by the Course ID. Thus, this example is in second normal form because the primary key consists of only one attribute (Course ID). (Hoffer, Ramesh, and Topi, 2011, pg. 186).

This user view is not in third normal form, however, because it contains the following transitive dependency:

Table 2-5. Transitive Dependency

COURSE ID →	Instructor First Name, Instructor Last Name

That is, if the instructor ID is dependent on the course ID, and the instructor first name and instructor last name are dependent on the instructor ID, then the instructor first name and instructor last name are dependent on the course ID, thus making it a transitive dependency.

Figure 2-5 shows the transitive dependency removed.

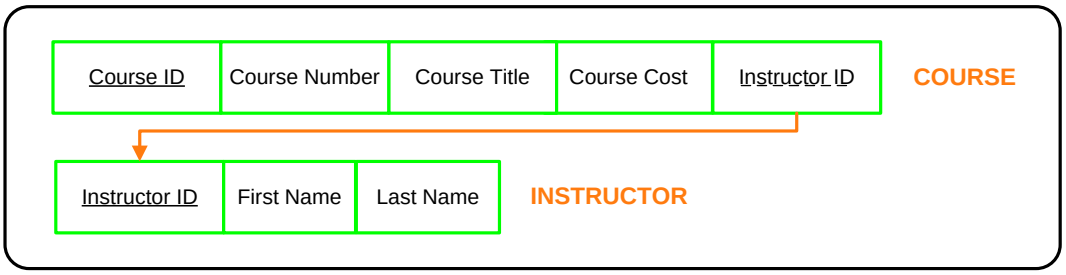

Figure 2-5. Transitive Dependency Removed (3NF)

The user view is now in third normal form.

Figure 2-6 shows the complete ERD with attributes shown and the primary and foreign keys defined.

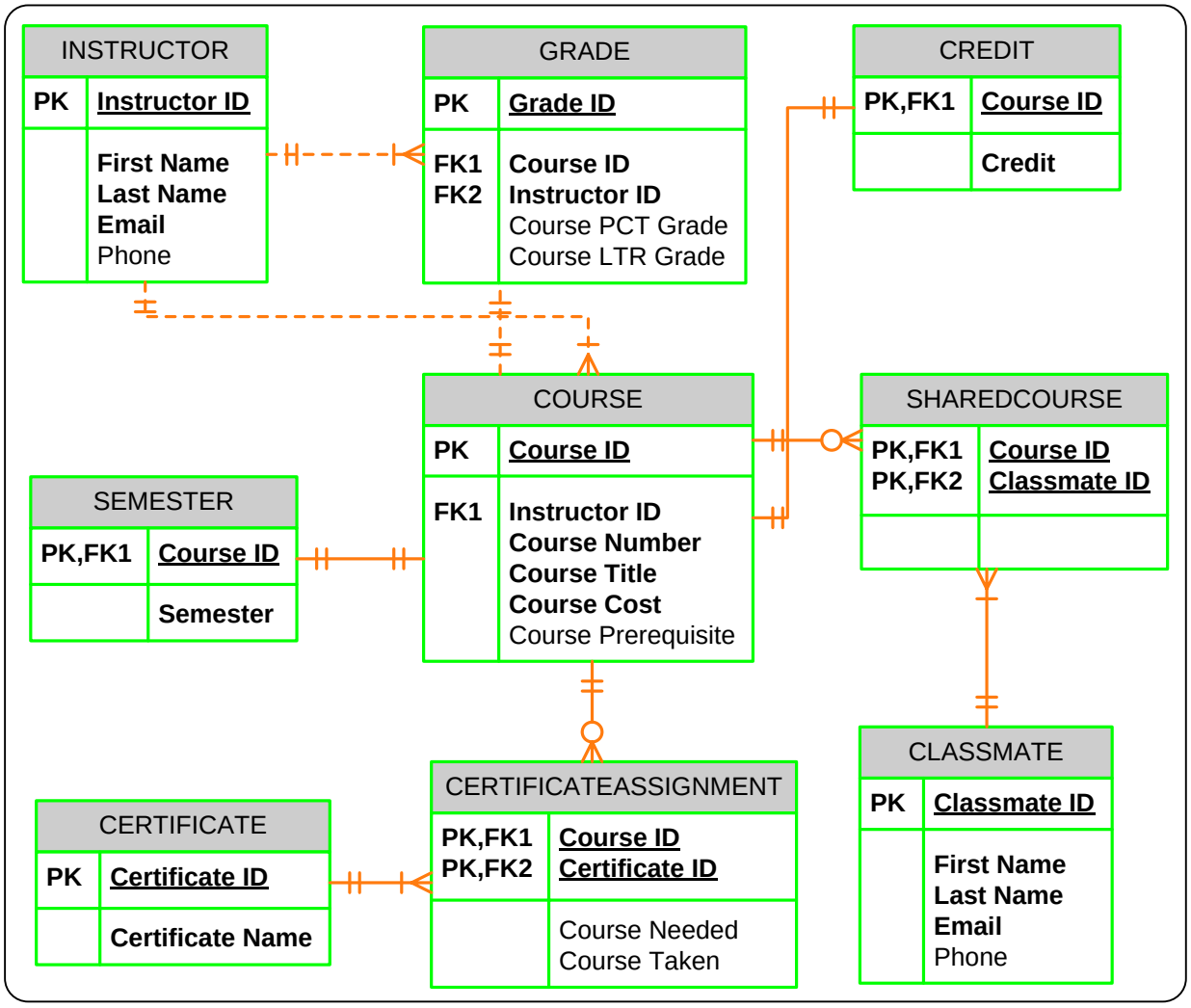

Figure 2-6. Complete ERD with Primary and Foreign Keys Identified

Section 3: Physical Design

Introduction

Up to this point, all references and diagrams for the database have referred to the logic behind the database. In this section, the diagrams and references will refer to the physical construction of the database, including naming conventions, data types, and the schema.

Naming Conventions

Entities

The text provides guidelines for naming entities (Hoffer, Ramesh, and Topi, 2011, Pgs. 70-71). Following those guidelines, the entity names and definitions found in Table 2-2. Entity Definitions were created. They are all singular nouns that are specific to me. They are all concise and used uniformly throughout the database.

Because this database is for my personal use, and will not be used or shared with others, the definitions are relatively simple and do not need to be complex in anticipation for future growth.

Attributes

Using the naming guidelines listed in the text (Hoffer, Ramesh, and Topi, 2011, Pgs. 76-77), the attributes listed in Table 2-2 were created. The attributes are further defined in Table 3-1.

Table 3-1. Attribute Definitions

ATTRIBUTE	DATA TYPE	DEFINITION	NULL VALUES
COURSE ID	Integer	The identifier for a course.	Not null
COURSE NUMBER	Char(10)	The number of a course given by NVCC; consists of a three-letter subject identifier and a three-digit number that specifies the relative difficulty of the course.	
COURSE TITLE	Varchar(50)	The name of a course given by NVCC.	
COURSE COST	Currency	The amount of a money a course costs.	
COURSE PREREQ	Char(10)	A course that must be taken prior to taking the course to which the prerequisite refers.	
INSTRUCTOR ID	Integer	The identifier for an instructor.	Not null
GRADEID	Integer	The identifier for a grade.	Not null
COURSE LTR GRADE	Char(1)	The grade received in a course as a letter (A through F).	
COURSE PCT GRADE	Decimal(3,1)	The grade received in a course as a percentage	
INSTR FIRST NAME	Varchar(30)	The first name of an instructor.	
INSTR LAST NAME	Varchar(30)	The last name of an instructor.	
INSTR EMAIL	Varchar(50)	The email address of an instructor.	

INSTR PHONE	Char(12)	The phone number of an instructor.	
CLASSMATE ID	Integer	The identifier for a classmate.	Not null
CLSMT FIRST NAME	Varchar(30)	The first name of a classmate.	
CLSMT LAST NAME	Varchar(30)	The last name of a classmate.	
CLSMT EMAIL	Varchar(50)	The email address of a classmate.	
CLSMT PHONE	Char(12)	The phone number of a classmate.	
CERTIFICATE ID	Integer	The identifier for a certificate.	Not null
CERT NAME	Varchar(50)	The name of a certificate.	
COURSE NEEDED	Bit	Specifies whether a course still needs to be taken. (Yes/No value)	
COURSE TAKEN	Bit	Specifies whether a course has been taken. (Yes/No value)	
CREDIT VAL	Integer	The number of credits a course is worth.	
SEMESTER DATE	Varchar(30)	The semester date of the course specified as: Term and Year. For example: Fall 2012	

If this database had the potential to be used by someone other than me, then some of the attribute abbreviations may have to be changed because they may not be intuitive (CLSMT for classmate). But, because I am the only one who ever is going to use this database, the attributes make sense to me, which is sufficient.

PHYSICAL NAME

The logical design of this database does not indicate how the attribute names will actually appear in the physical database. In the physical database, the names will not have spaces. For example, Course ID will actually be CourseID; Instructor ID will be InstructorID, and so on. They will use Pascal case.

Relationships

The relationships have been defined in the **Relationship Definitions** section. They follow the guidelines specified in the text. (Hoffer, Ramesh, and Topi, 2011, Pgs. 93-94.)

Table Names

The table names in my database will share names with their respective entities, with the addition of an underscore and a "T." So, the course table will be called COURSE_T; the instructor table will be called INSTRUCTOR_T, and so on.

Physical Schema

Figure 3-1 shows the physical schema with the names as they will appear in the database. Notice that the conventions described in the Physical Name and Table Name sections above appear in Figure 3-1.

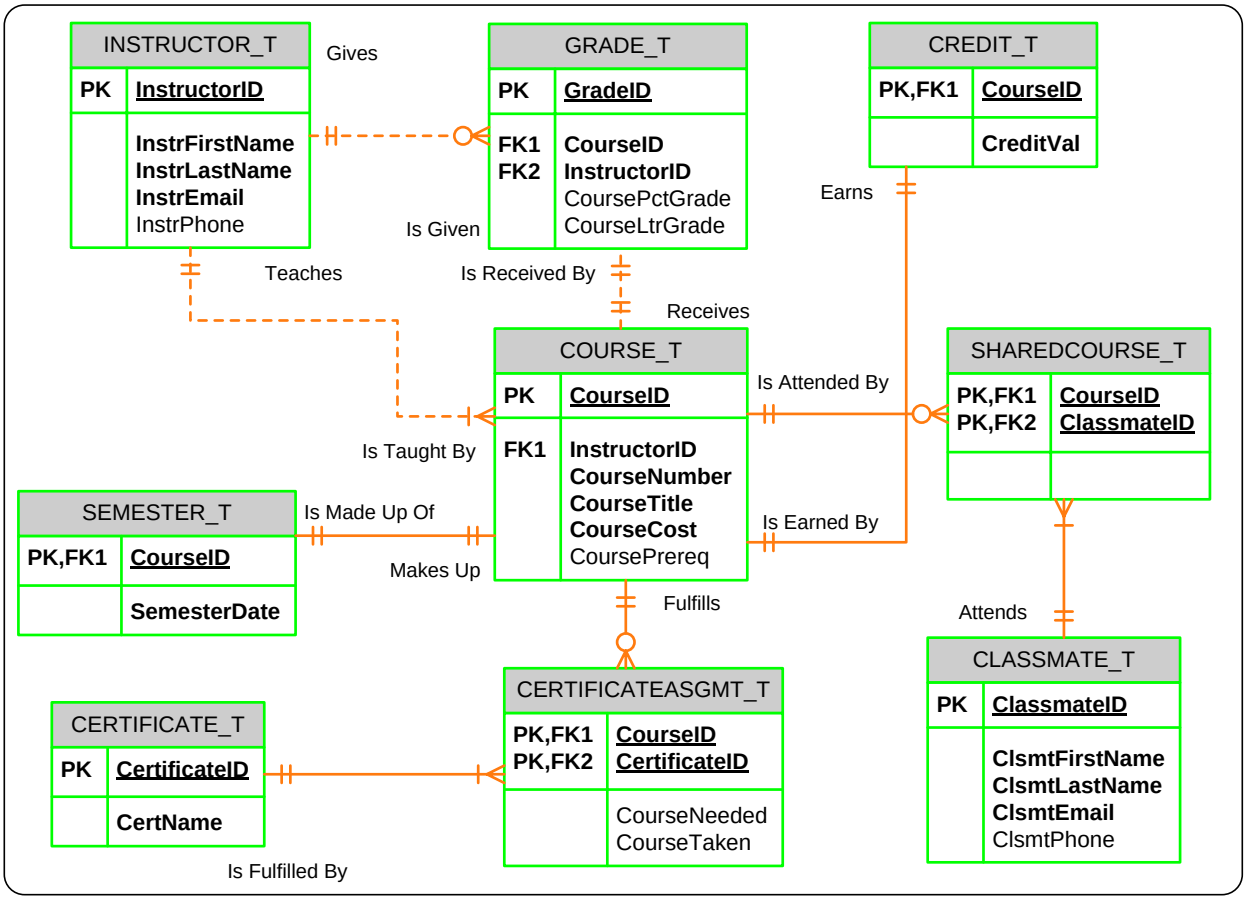

Figure 3-1. Physical Database Schema

Column Names

Table 3-2 shows the column names for each table. Notice that the column names are the same as the attribute names.

Table 3-2. Column Names

INSTRUCTOR_T	InstructorID, InstrFirstName, InstrLastName, InstrEmail, and InstrPhone
GRADE_T	GradeID, CourseID, InstructorID, CoursePctGrade, and CourseLtrGrad.
CREDIT_T	CourseID and CreditVal
SEMESTER_T	CourseID and SemesterDate
COURSE_T	CourseID, InstructorID, CourseNumber, CourseTitle, CourseCost, and CoursePrereq
SHAREDCOURSE_T	CourseID and ClassmateID
CERTIFICATE_T	CertificateID and CertName
CERTIFICATEASGMT_T	CourseID, CertificateID, CourseNeeded, and CourseTaken
CLASSMATE_T	ClassmateID, ClsmtFirstName, ClsmtLastName, ClsmtEmail, and ClsmtPhone

Data Types

Table 3-1 shows the data types for each attribute. Listed below are the reasons behind the data type selections that may seem counterintuitive.

- CourseNumber is char(10) because the course number will never be more than 10 characters. (CoursePrereq uses the same number, so it too will be char(10)).
- CoursePctGrade is decimal(3,1) because no percent grade will require more than three digits in the number or more than one digit to the right of the decimal.
- InstrPhone and ClsmtPhone are both char(12) because the data will be in the standard phone number format: XXX-XXX-XXXX.
- SemesterDate is not in date format because it will have the format term name, year. (Ex. Fall 2012).

Section 4: Design Revisions

Introduction

Per the feedback given, the following revisions have been made to the logical and physical design of the database to address the issues presented.

Issue 1

On page 8, in the course entity, you need to be able to distinguish a course by when it is taken. Here's why: In the off chance you must repeat a course, you will not be able to enter all occurrences of it because the course id is the primary key. This model assumes a course is taken once time. OR are you using this entity to simply inventory the courses you require. If so, this needs to be explained some place in your paper.

Solution 1

My company reimburses me for the completion of this degree. But, in order to receive reimbursement, I must receive at least a grade of C in a class. Also, I do not intend to have the need to take a class a second time because of the time and money that would be wasted if I did so.

With that said, the database should allow a course to be repeated without causing confusion. To address this concern, I have removed the CREDIT and the SEMESTER entities, and instead rolled them into the COURSE entity as attributes.

In the event that I need to take a course a second time, the second instance of the course will receive a new Course ID, even if the course is taught by the same instructor, because it will have a different semester date value. See **VIII. The** course ID will consist of the course name and the semester date..

Issue 2

What will the "Grade ID" look like? Is this going to be a surrogate key? Anything else might not work as a primary key.

Solution 2

Grade ID has been removed from the database. Originally, it was intended to be surrogate key but, after review, the composite key of Course ID and Instructor ID is sufficient in identifying the entity. (See **Business Rules** number 1, 2 and 3 – a course can receive one and only one

grade and a course can only have one and only one instructor.) There will never be an instance of two rows in the Grade table having the same Course ID and Instructor ID combination.

Even if the same course is taken with the same instructor a second time, it will have a different course ID (see **Solution 1**.)

The GRADE entity will also have a Completion Date attribute, which will contain the date the course was completed (when a grade was given by an instructor and received by the course).

Issue 3

You need to remove course id from instructor. If an instructor teaches multiple courses, you will only be allowed to enter one occurrence since this is the primary key. Use this entity solely for the name and contact info.

Solution 3

The Course ID attribute appeared in Table 2-1 as a foreign key in the INSTRUCTOR entity as an oversight. After working through the physical design of the database, I realized that leaving it in there would not allow an instructor to teach more than one course because there would be multiple values for it, if the instructor taught more than one course.

You will notice that in the rest of the figures and tables, it had been removed. The INSTRUCTOR entity consists of InstructorID, InstrFirstName, InstrLastName, InstrEmail, and InstrPhone.

Issue 4

Is Classmate ID a surrogate key?

Solution 4

ClassmateID is indeed a surrogate key because a classmate's first name and last name cannot serve as primary key in the off chance that two different classmates have the same first and last names, or in the event that both of these classmates have the same first and last names and are both in the same class with me.

Issue 5

Are you saying the shared course contains a composite primary key? Please note this on page 8.

Solution 5

The SHAREDCOURSE associative entity does have a composite key comprised of CourseID and ClassmateID. I thought I did denote that by underlining them both.

Issue 6

I don't think you need a separate entity for credit. This information can be added to the course entity if you are using the course entity as an inventory of information for all required courses [not courses completed].

Solution 6

The CREDIT entity has been removed. Instead, credit becomes an attribute that is part of the COURSE entity.

Issue 7

I don't see the value in the semester entity. HOWEVER, this raises the question of where do you track when a course was successfully completed and what was the grade?

Solution 7

The SEMESTER entity has also been scrapped, much like the CREDIT entity, as explained in Solution 6. This entity also becomes an attribute of the COURSE entity.

The successful completion of a course will be tracked in the GRADE entity. See Solution 2.

Issue 8

You must update your ERD descriptions. Explain at which attribute the relationship exists. [I need to know that you know this.]

Solution 8

COURSE ID

Course ID in the COURSE entity has a 1:1 relationship with the Course ID in the GRADE entity because a course can receive only one grade and a grade can be given to only course.

Course ID in the COURSE entity has a M:N relationship with the Course ID in the CLASSMATE entity because a course can have many classmates and a classmate can have many courses in common with me. The SHAREDCOURSE associative entity resolves this by associating a specific course (Course ID) with a specific classmate (Classmate ID).

Course ID in the COURSE entity has a M:N relationship with the Course ID in the CERTIFICATE entity because a course can fulfill one or more certificate requirements and a certificate requirement can be fulfilled by more than one course. The CERTIFICATEASSIGNMENT associative entity resolves this by associating a specific course (Course ID) with a specific certificate requirement (Certificate ID).

INSTRUCTOR ID

Instructor ID in the INSTRUCTOR entity has a 1:M relationship with the Instructor ID in the COURSE entity because an instructor can teach one or more courses and each course must be taught by one instructor.

Instructor ID in the INSTRUCTOR entity has a 1:M relationship with the Instructor ID in the GRADE entity because an instructor can assign one or more grades (depending on how many courses they teach) but a grade must have only one instructor assigning it.

CLASSMATE ID

Classmate ID in the CLASSMATE entity has a M:N relationship with the Classmate ID in the COURSE entity because a classmate can attend several courses with me and a course can have several classmates attend it. The SHAREDCOURSE associative entity resolves this by associating a specific course (Course ID) with a specific classmate (Classmate ID).

CERTIFICATE ID

Certificate ID in the CERTIFICATE entity has a M:N relationship with the Certificate ID in the COURSE entity because a certificate can be fulfill by one or more courses and a course can fulfill more than one certificate requirement. The CERTIFICATEASSIGNMENT associative entity resolves this by associating a specific course (Course ID) with a specific certificate requirement (Certificate ID).

Entity Relationship Diagram

The entity relationship diagram shown in Figure 4-1 has been updated to remove the SEMESTER and CREDIT entities. See **Solution 6** and **Solution 7**.

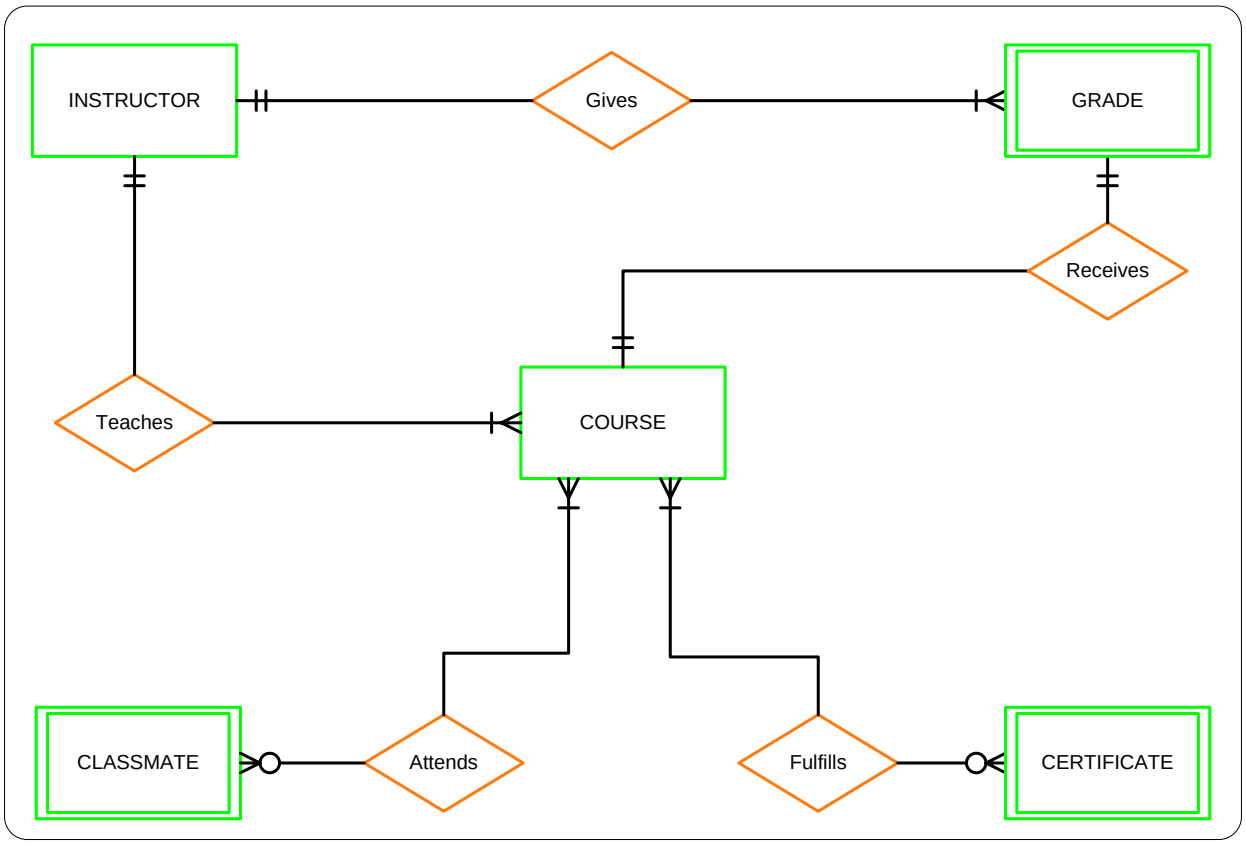

Figure 4-1. Entity Relationship Diagram Revised

Referential Integrity Diagram

The referential integrity diagram, shown in Figure 4-2, has been updated to remove the SEMESTER and CREDIT entities. See **Solution 6** and **Solution 7**.

The Grade ID attribute has also been removed from the GRADE entity, while the Completion Date attribute has been added. See **Solution 2**.

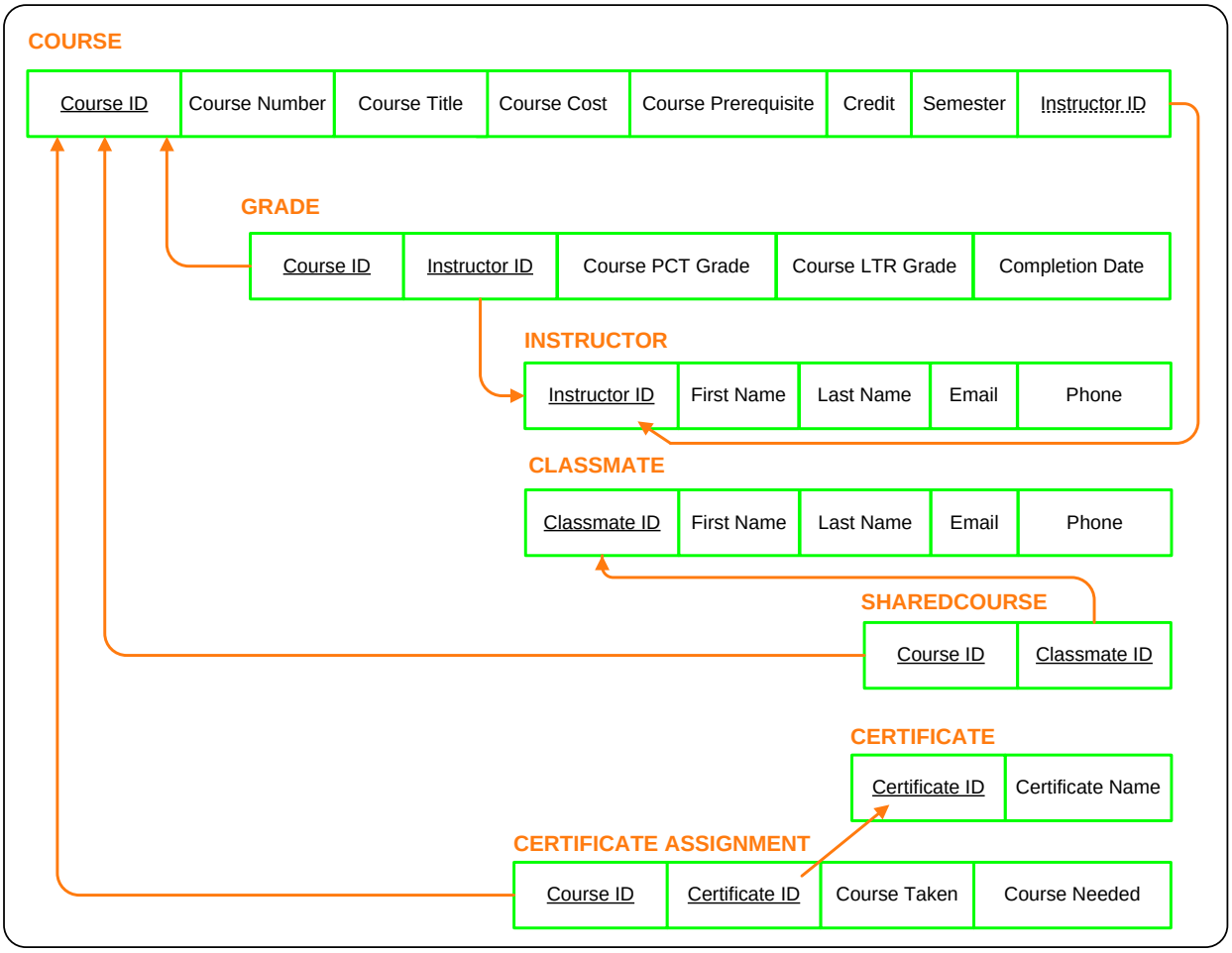

Figure 4-2. Referential Integrity Being Reinforced Revised

Complete ERD

The complete entity relationship diagram, shown in Figure 4-3, has been updated to remove the CREDIT and SEMESTER entities. See **Solution 6** and **Solution 7**.

The Grade ID attribute has also been removed from the GRADE entity, while the Completion Date attribute has been added. See **Solution 2**.

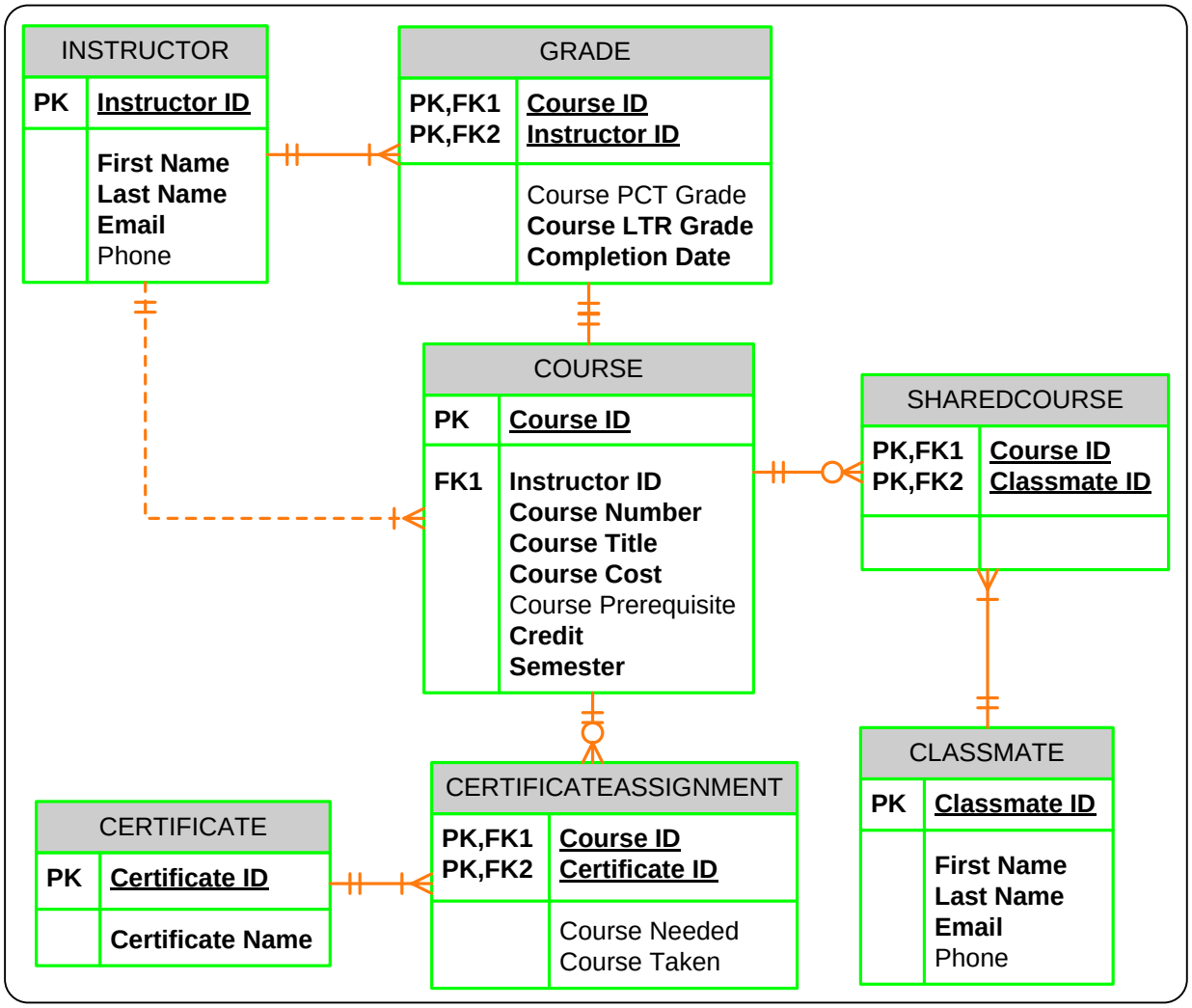

Figure 4-3 Complete ERD with Primary and Foreign Keys Identified Revised

Physical Schema Revised

The physical schema diagram, shown in Figure 4-4, has been updated to remove the CREDIT and SEMESTER entities. See **Solution 6** and **Solution 7**.

The Grade ID attribute has also been removed from the GRADE entity, while the Completion Date attribute has been added. See **Solution 2**.

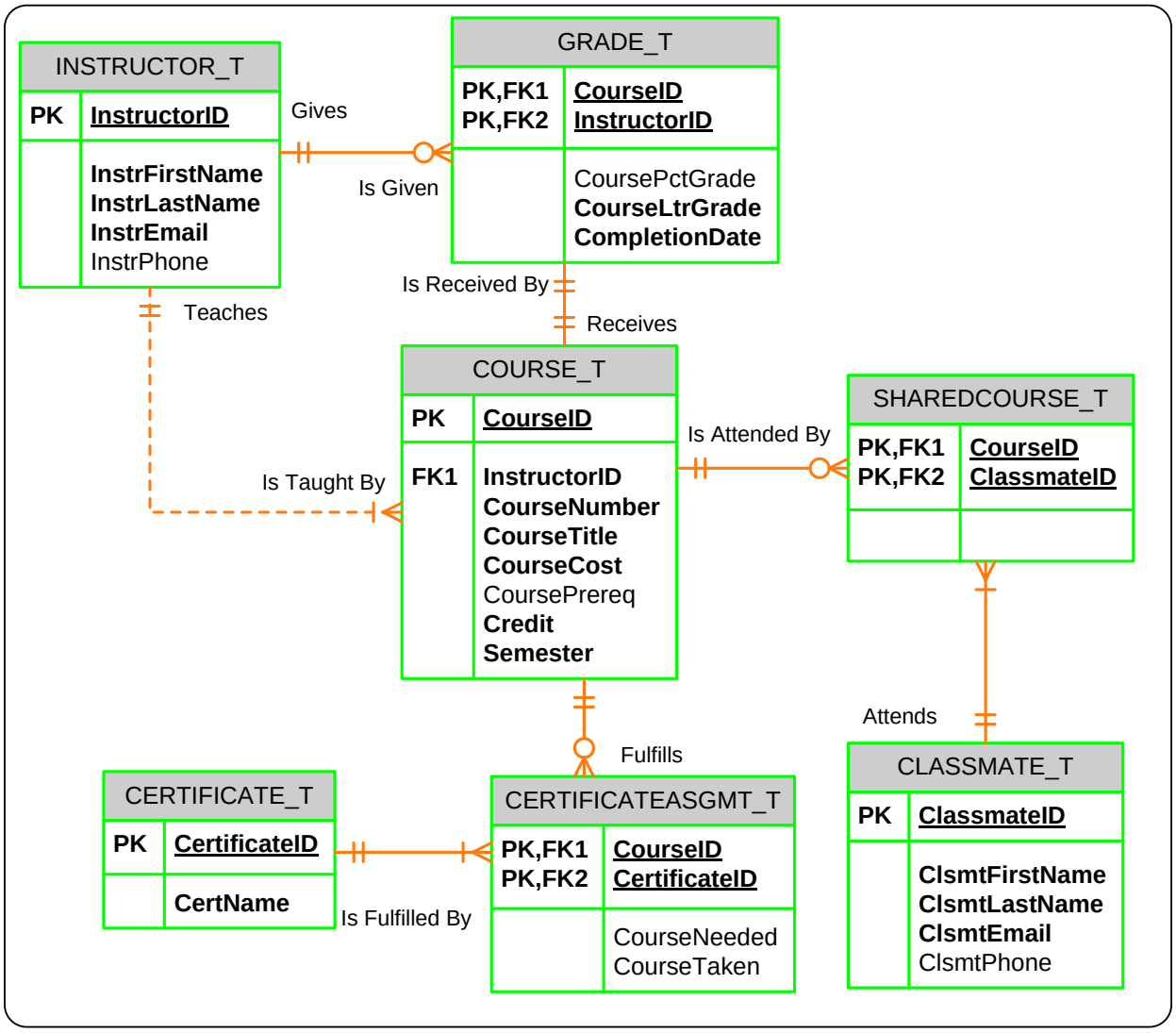

Figure 4-4. Physical Database Schema Revised

Business Rules Revised

The business rules stated her are in addition to the rules stated in the Business Rules section.

VIII. The course ID will consist of the course name and the semester date.

The course ID attribute will consist of a three-letter subject identifier, a three-digit number corresponding to the relative difficulty of the course, a two-letter semester name identifier, and a a two-digit year identifier.

Valid semester identifiers are FA (fall), SP (spring), and SU (summer.)

For example, a course ID would be ITN100-FA11 or ITE115-FA11.

Business Rules VI and VII no longer exist because the CREDIT and SEMESTER have changed from separate entities to attributes of the COURSE entity.

Section 5: Implementation

Introduction

With the issues corrected and the physical design complete, my college degree progress database moves to the implementation phase. I chose to use Microsoft Access 2010 as my database administration system. I contemplated choosing SQL, as I was excited to use it, considering I use it a little at my job.

Ultimately, I decided to use MS Access for the following reasons.

Familiarity

Much like I would if I were setting up a database for a businessfor which I worked, I chose a database management system I was familiar with, had used before, and felt comfortable using.

Having used MS Access before, I wouldn't have to worry about setting up the database incorrectly, or worry about attemping to learn a new technology while in the process, learning and understanding database concepts.

By choosing MS Access, I was allowed to focus on learning the database concepts, and a little SQL, without having to gain in-depth knowledge. This kept me from taking on too much.

Popularity

According to Blue Claw Database Design, MS Access is the most widely used desktop database management system in the world. Given its popularity, I knew I could readily and easily find help, if I wasn't sure how to perform a specific action. I could search Microsoft's help sections, knowledge bases, and forums. With that said, similar information is available with SQL, so this isn't really an advantage that using MS Access provides over SQL.

Simplicity

MS Access has similar functionality to other Microsoft products I've been using since high school, making it easier to get acclimated to MS Access.

Setting up a database that is as simple as my college degree database took about an hour, and wasn't very difficult.

Portability

Blue Claw Database Design says that MS Access allows databases to be ported to SQL Server or Oracle, though this is not easy. It also allows the database to be integrated with a website so that remote users can access the database.

Conclusion

Overall, using MS Access saved me time (because I already am familiar with it) and money (because I don't have to purchase Oracle or SQL Server).

If the database required more security or were accessed by several users, I would consider using Oracle or SQL. But, for a small, single-user database, MS Access was the most cost-effective solution.

SQL

The SQL statements that would be used to perform specific actions for this database are shown in Table 5- 1.

Table 5- 1. SQL Statements

To create the database in SQL, I would use the following statement.	CREATE SCHEMA Degree; AUTHORIZATION msmith;
This statement creates the Instructor_T table, with all of the attributes (columns) defined, as well as the primary key, data types and lengths.	CREATE TABLE Instructor_T (InstructorID VARCHAR(30) NOT NULL PRIMARY KEY, InstrFirstName VARCHAR(30), InstrLastName VARCHAR(30), InstrEmail VARCHAR(50), InstrPhone CHAR(12));
This statement creates the Course_T table with all of the columns defined and their data types, as well as which attributes cannot be null. It also specifies that CourseID is the primary key, while InstructorID is a foreign key.	CREATE TABLE Course_T (CourseID CHAR(11) NOT NULL PRIMARY KEY, CourseNumber CHAR(6), CourseTitle VARCHAR(50), CourseCost CURRENCY, CoursePrereq CHAR(6), Credit INTEGER, Semester CHAR(4), InstructorID VARCHAR(20) FOREIGN KEY REFERENCES Instructor_T (InstructorID));
To add, modify, or delete data meeting a certain condition, I would use the following statement. I used this syntax because it allows me to enter data where an attribute will be left null (CoursePrereq).	INSERT INTO Course_T (CourseID, CourseTitle, CourseCost, Credit, Semester, InstructorID) VALUES ('ITE115-FA11','ITE115','Intro to Computer Applications and Concepts', 413.25,3,'FA11','hserrato');

I could also use the following statement to insert the InstructorID into the Course_T table. This would insert Hugo Serrato's InstructorID (hserrato) into the Course_T table.	INSERT INTO Course_T SELECT InstructorID FROM Instructor_T WHERE InstrName = 'Hugo Serrato';
To create a view, I would use the following SQL statement. This creates a view that shows the course ID, course title, and instructor ID for all courses in the Course_T table where the semester is FA11 (Fall 2011).	CREATE VIEW [Fall 2011 Semester] AS SELECT CourseID, CourseTitle, InstructorID FROM Course_T WHERE Semester='FA11';
To delete a table, I would use the following SQL command.	DROP TABLE Course_T;

Section 6: Maintenance

Introduction

Once the database is implemented, the following security measures and procedures will be put into place to ensure that the database is meeting its primary goals, and that the authenticity of the data is maintained.

Security

For this particular database, no security is really required, as it does not contain sensitive information, or any information that is valuable to anyone but me. With that said, I may want to prevent people from being able to modify the contents.

While it is highly unlikely, my seven-year old daughter may accidentally open the database. I don't want her to have the ability to accidentally delete or modify the data. So, I would implement a password.

If the database were used by multiple users, this probably wouldn't be enough security. But, because it is only used by me, this should suffice.

Testing

Again, because this database is only of interest to, and will only be used by me, testing is not as crucial as it would be for a larger database used by others. If, for example, I wasn't the only one using it, then load testing, among other forms of testing, might be more important.

Still, I should test the database to make sure that it works properly and provides the correct information. To do this, I would write several queries to execute that would display the data in the tables, and verify that against my raw data, which is currently in Excel spreadsheets.

I would also check to make sure data can be added, modified, and deleted from the database properly. I would only conduct this testing if I added a table or made significant changes to the database itself.

Feedback/Updates

With this database being personal, there is really no need to come up with a system for providing feedback. I am the only one that needs to use it, so I would be able to keep track of what updates and changes are made.

One thing I would do is document the changes so that even if I don't use the database for several months or even years, I still know what I changed over time and the thought process behind my changes.

References

Blue Claw Database Design LLC. *Microsoft Access Database, Why Choose It?* Retrieved from http://www.blueclaw-db.com/microsoft_access.htm

Chapple, M. *Referential Integrity*. Retrieved from http://databases.about.com/cs/administration/g/refintegrity.htm

Singh, M. (January 2012) *Database Testing Basics - How to test and what to test?* Retrieved from http://www.softwaretestingtimes.com/2012/01/database-testing-how-to-test-and-what.html

Hoffer, J., Ramesh, V., & Topi, H. (2011). *Modern Database Management*. Saddle River, NJ: Prentice Hall

UBR, Inc. (2001). *What naming convention should I use in my database?* Retrieved from http://databases.aspfaq.com/database/what-naming-convention-should-i-use-in-my-database.html

W3Schools.com. *SQL Primary Key* Retrieved from http://www.w3schools.com/sql/sql_primarykey.asp

Wikipedia.org. *Functional Dependency* Retrieved from http://en.wikipedia.org/wiki/Functional_dependency

Yousaf, T. *Entity Relationship Diagram*. Retrieved from http://www.slideshare.net/tameemyousaf/entity-relationship-diagram-erd-5695380