# *Development Project II (420-E63-HR)*

# *Assignment 2 – Code inspection*

Date assigned: Tuesday, February 27, 2018

Date due: **Thursday, March 1, 2018, 11:50PM (No late assignments accepted)**

**Learning Objectives**

Upon successful completion of this assignment, the student will:

* Review and analyse the current database
* Clearly understand the purpose and function of the database, the tables, columns and relationships.
* Confirm that the database meets design, privacy and performance objectives.
* Propose any refactoring and determine whether any refactoring should proceed.

Note: we will not dive into security or business logic layer at this time. That will come later.

To do:

There are individual and team portions of this assignment. **YourUserName\_Teamxx\_E63\_A04\_DatabaseReview.docx.** Include your (short) username and team number. Much of this work is based on items covered in the Database I and II courses. You may need to review those courses.

# Section 1 – Individual Work

## Part A – Review and Analysis

1. Identify the entities in your database and how they are used.

Note: capture high level business rules related to the Entity (i.e. what does this entity represent from business perspective). Explain the business flow when each table has entries that are Created, Updated, Retrieved, Deleted

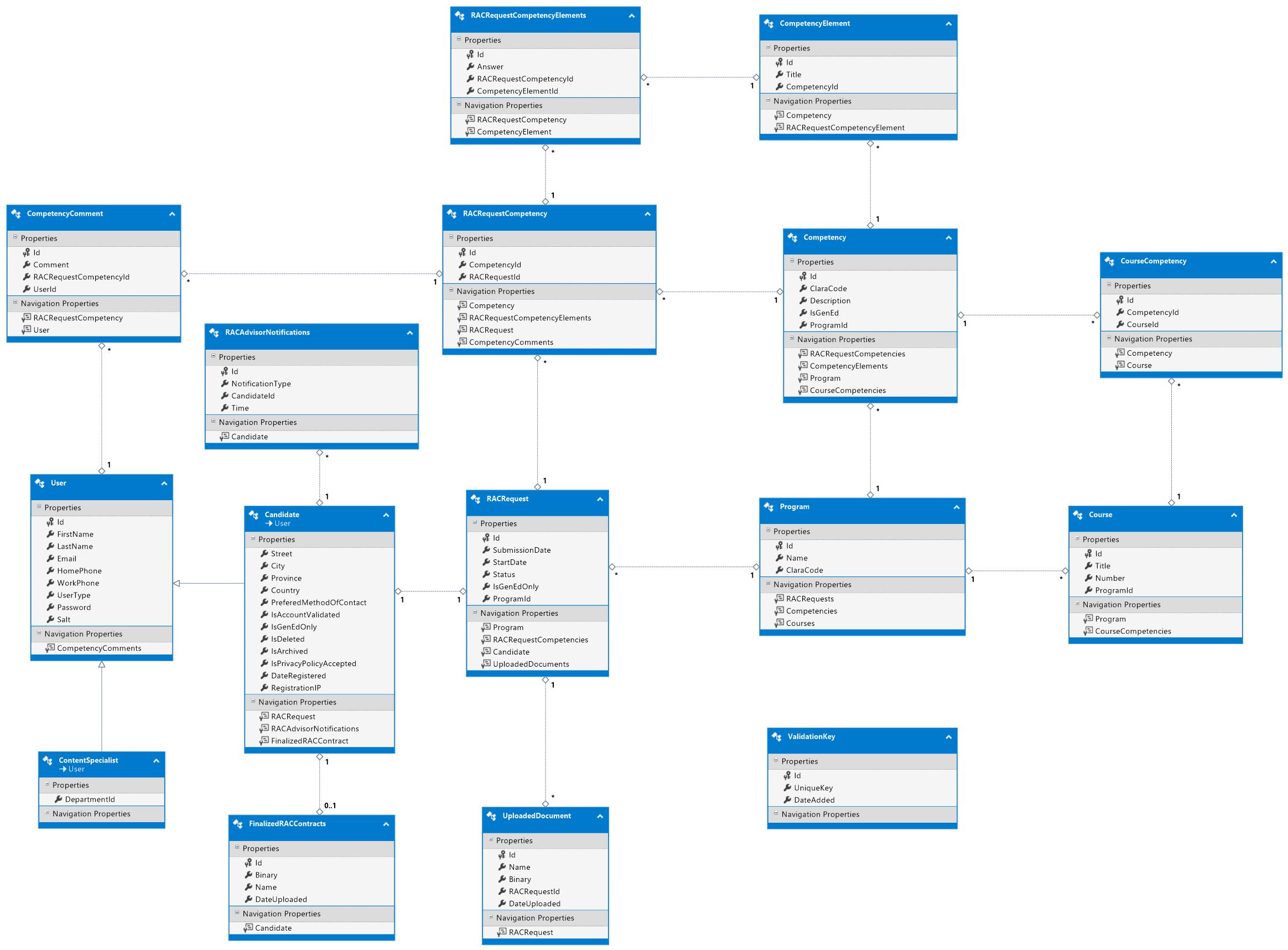
| **Entity Name** | **Physical Name** | **Definition, Description, Rules** | **Created** | **Updated** | **Retrieved** | **Deleted** |
| --- | --- | --- | --- | --- | --- | --- |
| Competencies | [dbo].[Competencies] | This table represents the competencies that exist within a program. They’re created based off of “Objectif”s that exist in Clara | When a competency is added to an active program, the active program gets duplicated with the new competency and the old one becomes inactive | Don’t get updated | A Competency is retrieved through it’s associated Program or RAC Request Competency | Don’t get deleted |
| CompetencyElements | [dbo].[CompetencyElements] | Represents the performance criteria for a given competency. | When a new competency element gets created the same happens as for competency | Don’t get updated | A Competency Element is retrieved through it’s associated competency or RAC Request Competency Elements | Don’t get deleted |
| Courses | [dbo].[Courses] | Represents a course in a given program. Comes from the “Cours” table in Clara. | When a new course gets created the same happens as for competency | Don’t get updated | Course are retrieved through their given program or from a competency associated to that course | Don’t get deleted |
| FinalizedRACContracts | [dbo].[FinalizedRACContracts] | Stores PDFs of the signed contracts between the RAC Advisor and the Candidate. | The documents uploaded are just plain PDFs, nothing notable happens. | Don’t get updated | Nothing notable happens, they’re just retrieved and downloaded to the viewer’s computer. | Don’t get deleted |
| Programs | [dbo].[Programs] | Stores all the programs RAC offers; These are pulled from Clara | When a new program gets added it will remain inactive until the RAC advisor activates it. | Whenever a program is updated, the program is duplicated with the old one marked as inactive | Retrieved from the RAC request | Don’t get deleted |
| RACAdvisorNotifications | [dbo].[RACAdvisorNotifications] | They’re only stored short term and are viewed by the RAC Advisor | Created whenever the user creates an account, validates it, changes program, etc | Don’t get updated | They’re viewed by the RAC Advisor |  |
| RACRequest | [dbo].[RACRequest] | Created for a user and is always tied to them. Links documents, programs and user self-evaluations on competencies. | Whenever a RAC Request is added it is tied to a User and they are stuck with it. | When it gets update the only things that really change are the program IDs. When the Program ID changes, a new entry is created in RACAdvisor Notifications | Retrieved through the Candidate using the foreign key in the Candidate | Don’t get deleted |
| RACRequestCompetency | [dbo].[RACRequestCompetency] | Contain the elements of the competency for the user to complete their self-evaluation | Links to a RACRequest | Nothing happens when it’s updated | Retrieved through the RAC Request | Don’t get deleted |
| RACRequestCompetencyComment | [dbo].[RACRequestCompetencyComment] | Allows the user to justify their self-evaluation in comments, and for the RAC Advisor and Cont. Spec. can comment on their evaluation. | Created by the Candidate, RAC Advisor and the Content Specialist | Nothing really happens when it’s updated | Retrieved through the RACRequest Competency | Don’t get deleted |
| RACRequestCompetencyElements | [dbo].[RACRequestCompetencyElements] | Stores the users answer for each of the different elements of a competency | Created when a new RAC Request Competency is created | Nothing really happens when it’s updated | Retrieved through the RACRequest Competency | Don’t get deleted |
| UploadedDocuments | [dbo].[UploadedDocuments] | Stores the documents that have been uploaded by a candidate. Don’t get deleted, only hidden. | Created whenever a user uploads a document to the system. | Don’t get updated | Nothing notable happens, they’re just retrieved and downloaded to the viewer’s computer. | Don’t get deleted |
| User | [dbo].[User] | Parent class that stores the information for candidates and content specialists |  |  |  | Don’t get deleted |
| User\_Candidate | [dbo].[User\_Candidate] | Inherits from User. Don’t get deleted, get flagged to be hidden |  |  |  | Don’t get deleted |
| User\_ContentSpecialist | [dbo].[User\_ContentSpecialist] | Inherits from user. |  |  |  | Don’t get deleted |
| ValidationKeys | [dbo].[ValidationKeys] | Allows the system to send emails to users to validate their accounts. |  | Don’t get updated |  | They get deleted |

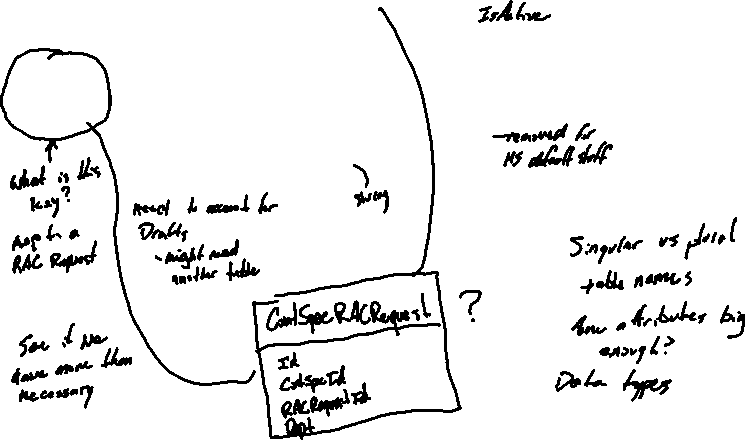
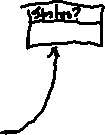
1. Identify the relationships, the meaning of the relationship and the cardinality

Relationship Definitions

| **First Entity** | **Second Entity** | **Relationship Name** | **Cardinality** | | **Relationship Definition** |
| --- | --- | --- | --- | --- | --- |
| **Min[[1]](#footnote-1)** | **Max** |  |
| User | *CompetencyComment* | *UserCompetencyComment* | *1* | *Many* | *One user can have many comments* |
| RACRequestCompetency | CompetencyComment | RACRequestCompetencyCompetencyComment | Many | 1 | A competency has many comments associated to it |
| RACAdvisorNotifications | Candidate | RACAdvisorNotificationsCandidate | Many | 1 | A candidate can give the RAC Advisor many notifications |
| Candidate | FinalizedRACContracts | CandidateFinalizedRACContracts | 1 | 0/1 | A candidate can have a finalized contract |
| Candidate | RACRequest | CandidateRACRequest | 1 | 1 | A candidate will have a single RAC Request |
| RACRequestCompetencyElements | RACRequestCompetency | RACRequestCompetencyElementsRACRequestCompetency | Many | 1 | A single competency has many elements associated to it for a user to evaluate themselves on |
| RACRequestCompetency | RACRequest | RACRequestCompetencyRACRequest | Many | 1 | A RAC Request has many competencies associated with it for comments |
| RACRequest | UploadedDocuments | RACRequestUploadedDocument | 1 | Many | A RAC Request has many documents uploaded associated to it |
| Program | RACRequest | ProgramRACRequest | 1 | Many | Any number of users can make a RAC Request for a given program, but only one. |
| CompetencyElements | RACRequestCompetencyElements | CompetencyElementsRACRequestCompetencyElements | Many | 1 | Every competency element has many users that can evaluate themselves on it |
| CompetencyElements | Competency | CompetencyElementsCompetency | Many | 1 | Each competency has many elements associated to it |
| Competency | Program | CompetencyProgram | Many | 1 | Each program has multiple competencies associated to it |
| CourseCompetency | Competency | CourseCompetencyCompetency | Many | 1 | The CourseCompetency table is associative, so there’s many competencies |
| CourseCompetency | Course | CourseCompetencyCourse | Many | 1 | The CourseCompetency table is associative, so there’s many courses |
| Course | Program | CourseProgram | Many | 1 | Each program has many courses |

1. Capture the physical model (paste below).





1. Analyze the physical model and complete the inspection table below:

| **Check** | **Details/Violations** |
| --- | --- |
|
|  |
| Table names follow naming standard | *All Tales follow the proper naming standard: They’re all listed in UpperCamelCase with RAC always being capitalized* |
| Attributes follow naming standard | The IDs of all tables are named Id, all foreign keys are named TableNameId, and they’re all in UpperCamelCase |
| Attribute types are sufficiently sized, appropriate and consistent through database and application use | Boolean attributes are named IsCondition  All Ids are integers  All documents are stored as Binaries  Name and titles are stored as Strings  Dates/Times are stored as DateTimes  All follow the consistent naming and the datatype issues have been fixed. |
| Attributes are atomic (not composite), explicit (not derived), singular (not multi-valued), | All the attributes are atomic, explicit and singular in our database. |
| Privacy: The personal data stored is strictly required for the business case. | There’s no data that we store in our database for the Candidates that’s unnecessary. We give them multiple different options for contacting them, and have the minimum information required mailing them. |
| The only access to Clara is via the Clara’s ReportClient views. | N/A – Currently don’t access Clara |
| Readability/portability to Clara (or any other external db) access with synonyms (saves you from [server].[schema]. naming) | Clara not accessed right now.  CSAdmin accessed through an context model |
| Database is in 3rd normal form. (data integrity, no data is stored multiple times) | The database is in 3rd normal form: We have no many-many relationships, they’re all replaced with associative entities. All traps have been solved, and all the data is stored once and accessed with foreign keys to the rows that store it. |
| Database has referential integrity and constraints to enforce it. (keys can’t point to invalid entries). | No keys point to non-existent data |
| Keys have uniqueness constraints and guaranteed unique. | The primary keys in each of the tables are created as proper Key Identities |
| Keys are surrogate keys (system generated). Identify any nay natural keys (keys with business meaning or come from the outside world) | The keys start at 1 and are incremented by 1. Their values are system generated. |
| Versioning mechanism to ensure that the application and database is compatible. | In progress |

1. Analyze the stored procedures, functions and triggers. Identify the Stored procedures, functions and trigger. For each, determine when it is used from a business flow point of view.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name (stored proc, function or trigger)** | **Purpose** | **Usage (Business flows)** | **CRUD logic/validation only (no business logic) (Y/N)** |
| N/A | N/A | N/A | N/A |

1. Recommendations, refactoring. Based on your analysis, what needs to be fixed/changed. Capture these as you would a bug with appropriate titles, details and reasoning.

|  |  |  |
| --- | --- | --- |
| Item # | Title | Description/ details |
| 1 | Historic tables | We should have historic tables for old versions of programs, competencies and competency elements |
| 2 | No duplication of competency names | Create a table for all competencies that get added and just make references to it from current and historic tables, so the descriptions don’t get repeated. |
| 3 | No duplication of course name | Create a table for all courses that get added and just make references to it from current and historic tables, so the titles don’t get repeated. |
| 4 | No duplication of competency elements | Create a table for all competency elements that get added and just make references to it from current and historic tables so the descriptions don’t get repeated. |

# Section 2 – Team work

Book a team meeting with Richard. Sessions start at 8am, Friday for 90 minutes (hopefully it’s shorter, but book 90 minutes).

The agenda is as follows:

1. Review of entities, relationships
2. Review of physical model
3. Review of programmability (stored procs/functions, triggers)
4. Identify key business work flows
5. Exercise business work flows through the database
6. Identify bugs, and triage into backlog.

Throughout the meeting, decisions and actions will be made/taken. These are captured in the meeting minutes and the Actions Register (spreadsheet).

Bring your completed assignment from this assignment, Section 1 with you.

**To submit**

When you have completed the individual assignment, upload the **YourUserName\_Teamxx\_E63\_A04\_DatabaseReview.docx** document to Moodle.

See Moodle for marking Rubric.

1. Optionality (min = 0) [↑](#footnote-ref-1)