NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Creating a Knowledge Space for the SQL domain

Ana Sofia Coimbra José

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

CREATING A KNOWLEDGE SPACE FOR THE SQL DOMAIN

by

Ana Sofia Coimbra José

Dissertation report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

Advisor: Roberto Henriques

Co-advisor: Jorge Neves

November 2021

DEDICATION

I would like to dedicate this work to my parents, without whom I would have never got here. They have been a constant safe arbour and support in my life and they have provided me with every oportunity to keep learning and growing, both personally and academically. This appreciation is extended to my brother and remainder family.

ACKNOWLEDGEMENTS

A special word to my friends who have been my partners in crime during these years at Nova IMS. Without you this journey would have been much harder, lonelier, and definitely not so fun. We have all been in the same boat for a while now, from now on it might not be the same, but they will certainly keep sailing close together.

To all my friends who were always available to hear me talking about my thesis and my distresses, even when they did not know much about the subject, a huge thank you. Thank you also for the times of distraction. Everything needs balance, and life cannot only be about work. Your presence and support kept me going during these months.

For the help given in making available some pieces of literature in the subject of the thesis, I want to thank the personnel of the Nova IMS Library.

For his time and support in the last phase of the thesis, I want to thank professor Roberto Henriques.

At last, but definitely not the least, a huge thank you to professor Jorge Neves. This appreciation note does not only concern this thesis, but all my path at Nova IMS. We started as student and teacher in my bachelor and went to colleagues and now friends. It was because of the course we teach together that this thesis subject came up and without him, it would have never been possible to deliver this work. Besides the role of an advisor, he took the role of an expert in the domain and was present in every step of the way. Thank you so much for the tireless support.

ABSTRACT

Structuring the existing knowledge of a domain is a challenge. The domain items must be identified, and the precedencies between these items must be discovered. Fundaments of the Knowledge Space Theory (KST) were applied to create a Knowledge Space in a restricted part of the SQL programming language domain – going from the most basic SQL knowledge to operations like Deleting, Inserting and Updating records in the database. Four methodologies were introduced, but only one was chosen. The implemented methodology is based on an extension of KST, called Competence-Based KST. When compared to the alternatives, it implicates less time from the experts, it provides a combination of skills and domain tasks and its outcome is less subjective. The achieved Knowledge Space is, in fact, a Competence Space, made of subsets of competencies a person must master to have complete knowledge on the domain. Associated with the Competence Space, a Performance structure was also created, and the combination of both structures is a good basis for a more informative knowledge assessment and for guiding the professors in their teaching method. The result presented is a subset of the complete Competence Space, but from it, the entire space can be generated. An intermediary result was also found useful to help professors guaranteeing that all the content to test is being covered in the questions presented to the students.

KEYWORDS

Knowledge Space Theory; Competence-Based Knowledge Space Theory; SQL Server; Competence Space

INDEX

Table of Contents

1. Introduction 1
2. Literature Review 4
   - 2.1. Knowledge Space Theory Explained 4
   - 2.2. Knowledge Assessment Output using KST 6
   - 2.3. Knowledge Spaces vs Learning Spaces 6
   - 2.4. Closure Under Intersection 7
   - 2.5. How To Find The Items and Instances Of a Domain 9
   - 2.6. How To Go From The Individual Nodes To a Knowledge Structure 11
     - 2.6.1. Querying Experts 12
     - 2.6.2. Data Analytics in Students' Responses 14
     - 2.6.3. Analysis of Problem-Solving Processes 14
     - 2.6.4. Competence-Based KST approach 15
   - 2.7. Types of Questions in Knowledge Assessments 18
   - 2.8. How to Visualize and Store a Knowledge Space 19
3. Methodology 21
   - 3.1. Methods Comparison and Discussion 21
   - 3.2. How to Allow the Existence of Different Strategies to Solve a Task 23
   - 3.3. Identify the Competencies in the Domain 26
   - 3.4. Identify the Typical Tasks in the Domain 27
     - 3.4.1. Types of Questions 28
     - 3.4.2. How To Compare Two SQL Queries 29
     - 3.4.3. Tasks Creation 33
   - 3.5. Assign Competencies to the Tasks 34
   - 3.6. Inference Precedencies and Analyze the Results 35
   - 3.7. Derive the Base of the Competence Space 36
   - 3.8. Derive the Performance Structure Associated to the Base 36
   - 3.9. Visualize the Base of the Competence Space 37
4. Results and Discussion 38
5. Conclusions 41
6. Future Work and Limitations 42
7. Bibliography 44
8. Appendix A 49

9. Appendix B 53
10. Appendix C 69
11. Appendix D 72
12. Appendix E 76

LIST OF FIGURES

Figure 1 - KST Concepts' Hierarchy 4
Figure 2 - Precedence Diagram for the Six Types of Algebra Problems presented in Table 1 5
Figure 3 - Graph of the Knowledge Space $K1$ 8
Figure 4 - Graph of the Knowledge Space $K2$ 9
Figure 5 - Example of CBKST Competence and Performance Structures 17
Figure 6 - Methodology Diagram – Steps to Follow 22
Figure 7 - Example of Question Schema 31
Figure 8 - Solutions Schema 32
Figure 9 - Visualization of the Base of the Competence Space (overview) 38
Figure 10 - Visualization of the Base of the Competence Space (higher resolution) 76

LIST OF TABLES

Table 1 - Six Types of Problems in Elementary Algebra 5
Table 2 - Five Problems in High School Mathematics 8
Table 3 - The Six Categories of Cognitive Processes 11
Table 4 - Example of CBKST Task-Competency Matrix 16
Table 5 - Example of CBKST Task-Competency Matrix with Alternative Strategies 23
Table 6 - Alternative Strategies to Solve **[Task1]** 25
Table 7 – Description of the Attributes in the Solutions Schema 32
Table 8 - Matrix Numerical Insights 35
Table 9 - Identified Competencies 49
Table 10 - Identified Tasks (Open response) 53
Table 11 - Identified Tasks (Multiple-Choice) 65
Table 12 - Task-Competency Matrix 69
Table 13 - Base states of the Competence Space and associated Performance states 72

LIST OF ABBREVIATIONS AND ACRONYMS

KST	Knowledge Space Theory
CBKST	Competence Based Knowledge Space Theory
SQL	Structured Query Language
ITS	Intelligent Tutoring Systems
RDBMS	Relational Database Management Systems

1. INTRODUCTION

Dividing the knowledge of a domain in smaller items of knowledge and finding how these items relate with each other is an important task in many ways. The discovery of these kind of structures can not only help professors in their teaching approaches but also in understanding what the students really know, by evaluating them using more informative knowledge assessments based in the knowledge structure of the domain.

Assessing students' knowledge is crucial to measure the evolution and performance of students and plan next steps towards the improvement of learning and teaching (OECD, 2013). However, most tests used nowadays to this intent deliver a pure numerical evaluation – regardless of the type of questions asked, the evaluation results in a single number (Falmagne et al., 2006). This numerical value is expected to quantify the students' knowledge regarding the content covered, but this result cannot answer the question "What does the student know?", as it is only a number. In the best-case scenario, the answer to the question "How much does the student know?" is revealed. Altough, even that might not be accurate, depending on the type of questions used and the probability of lucky guesses (Burton, 2002; Bush, 2006). This is still a prevalent approach in many important and decisive knowledge assessment tests, such as the SAT[1].

Delivering results that are not accurate and do not truly reflect the students' knowledge in a particular domain can have a significant impact in their future by blocking access to opportunities or giving better opportunities to the ones who do not deserve them. Additionally, it can negatively impact the students' self-esteem (Reay & Wiliam, 1999). Some students face great difficulties to achieve good grades, and Clarke et al. (2003) found that most teachers consider that many students feel that, no matter how hard they try, they will still do poorly on a test. A knowledge assessment should guide the student on what to learn next and how they can improve, but a number does not give enough information to provide this guidance. It is imperative to start looking at knowledge assessment differently.

Knowledge assessment is not only about understanding what the students know. This is its primary goal, but while the students are being assessed, they can also be learning. How? There are two answers to this question, through feedback and through the presentation of problems that the student cannot solve yet. Kornell et al. (2009) found that accompanying wrong answers with feedback results in enhanced learning and concluded it was more efficient in the long run than spending the same time studying the answer. The same paper states that students learn more when studying a particular topic if they were previously asked about it, even if their first answer is incorrect, which is expected once they just entered the topic.

In the past decades, with the increase in the number of individuals that use the internet, from 0.049% in 1990 to 48.997% in 2017[2], new online-based learning systems were created and they have helped making education more accessible and convenient (Liu et al., 2020).

Intelligent Tutoring Systems (ITS) are an online-based learning system and refer to educational software containing an artificial-intelligence component (Shute & Zapata-Rivera, 2010). These systems

[1] The SAT is a standardized test widely used for college admissions in the United States.

[2] Source: https://data.worldbank.org/

track the work performed by the students and can suggest additional work. They can be used as a complement to in-class teaching, helping the teachers tracking students' performance and guiding the students on what to learn next, or as an autonomous way of learning. Although research in the ITS domain has been growing rapidly in the last few years, it is still in an early phase with much potential to be revealed (Guo et al., 2021).

According to Hartley & Sleeman (1973), ITS's must have knowledge of the learner, of the domain and of teaching strategies. An informative knowledge assessment is a crucial component of these systems. It is through knowledge assessment that these systems track the students' progress along the way (knowledge of the learner). To perform the knowledge assessment, it is necessary for the system to know what the student must master to have complete knowledge in the domain (knowledge of the domain). When the system has information on the current knowledge of the student it must be able to suggest the task to be asked/learnt next (teaching strategies). The system should be constantly updating the student's current knowledge and repeating the cycle described above.

This takes us back to the first paragraph of this section. For these systems to know what the student must master to have complete knowledge in the domain, the items of knowledge in the domain must be previously identified, and to point the students to the next task to perform these items must be organized in some way.

Doroudi et al. (2016) suggest that student learning can be sensitive to temporal sequencing, which may represent a hint for an alternative method of knowledge assessment. Is it possible to identify and store the knowledge of a specific domain, understand the temporal sequencing of its many topics, navigate through this "map", and deliver the student a more significant output than a number? The Knowledge Space Theory, introduced in 1985 by Jean-Paul Doignon and Jean-Claude Falmagne, gives a theoretical framework for creating a method capable of addressing these questions. As described by Falmagne et al. (2006)

"The basic concept of this theory is the 'knowledge state', which is the complete set of problems that an individual is capable of solving in a particular topic, such as Arithmetic or Elementary Algebra. The task of the assessor—which is always a computer—consists in uncovering the particular state of the student being assessed, among all the feasible states."

As opposed to the pure numerical approach, the output of this alternative way of assessing knowledge is not a number but a knowledge state. This delivers answers to two questions: "What does the student know" and "What is the student ready to learn next".

The objective of this thesis is to create the "map" mentioned above. It will constitute the basis for a knowledge assessment tool in the Relational Databases domain, particularly for a subset of the SQL language topic. To do so, the Knowledge Space Theory and its concepts will be studied and applied, enabling the creation of a knowledge structure for the domain. In the literature reviewed it was not found any application of the Knowledge Space Theory in the domain of Relational Databases.

Regarding the domain under study, a clarification is of importance. Not all the SQL programming language domain is to be considered for the sake of complexity containment. The domain boundaries go from the most basic SQL knowledge to operations like Deleting, Inserting and Updating records in the database. DDL operations (Create, Drop, Alter) are out of the scope of the domain, as well as

triggers, views, and other more advanced subjects of this programming language. Multiple database systems use SQL, like MySQL, PostgreSQL, Oracle RDBMS and Microsoft SQL Server, and some differences do exist between those systems, particularly in terms of code syntax. For this reason and to be as specific as possible, the domain refers to SQL programming language in a Microsoft SQL Server context (Transact SQL).

In what concerns the structure of this document: This section presented a context on the current state of students' knowledge assessment and evidenced the need of creating a structure that represents and stores the knowledge contained in a domain; In section 2, the KST and its concepts are introduced, a critical literature review that explores previous and related work to KST and typically used question types in knowledge assessment is presented and four possible methodologies to follow are introduced; Section 3 starts with a discussion on the approach to follow and after the decision is made all the steps taken to achieve the final knowledge space in the domain under study are described; In section 4, the achieved knowledge space and its validity is discussed based on a visual representation; Section 5 sums up the work done and presents the conclusions; Finally, in section 6, the limitations of this work are enumerated and future work is suggested.

2. LITERATURE REVIEW

This chapter aims to introduce the conceptual framework of Knowledge Space Theory (KST) and understand how it can be applied to create a knowledge structure in the SQL domain.

2.1. KNOWLEDGE SPACE THEORY EXPLAINED

In 1985, Jean-Paul Doignon and Jean-Claude Falmagne started developing a theory that would enable the modelling of students' knowledge. This new way of modelling knowledge made it possible to perform knowledge assessments that are more adaptive, taking into account the previous responses given by the student, and produce an outcome that is much more informative than the pure numerical approach (Doignon & Falmagne, 2016).

The core of this theory is the concept of knowledge state, which is the complete set of problems that an individual can solve in a particular topic. A knowledge structure is a collection of knowledge states. Types of problems are the units of knowledge (Doignon & Falmagne, 2016), also referred to as items (Falmagne et al., 2006). The latter will be adopted in the remainder of this document. In Beginning Algebra, a possible item would be "Integer addition", and the exercises corresponding to an item are called instances. An instance of the item "Integer addition" could be "What is the result of 27 + 195?" and all instances inside an item should have the same difficulty level (Falmagne et al., 2006). Figure 1 evidences the hierarchy between all the concepts mentioned above.

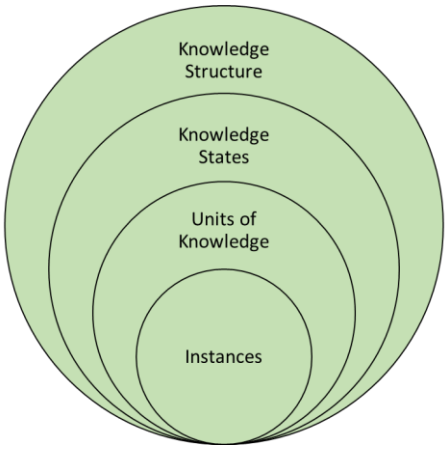


Figure 1 - KST Concepts' Hierarchy

KST assumes that every knowledge domain can be represented in terms of a set of domain items (Stahl & Hockemeyer, 2019), and it is based on the precedence relation, which is derived from the Learning Hierarchy Technique, first developed in 1962 by Gagné (Nwaogu, 2013). Gagné considered this technique as a teaching and learning tool that may contribute to greater effectiveness in engineering education, and he proposed that the learning process should be divided into multiple measurable objectives. One (1) terminal objective and the enabling objectives that lead to it. These enabling objectives would have their own enabling objectives until the ones found seemed simple enough to be performed by the students (Alias & Gray, 2005). In KST, hierarchical relations are found between the items, creating a structure that allows conclusions of the type: to master item *a*, the student must first master a set of other items that are its prerequisites.

The precedence relations can be seen through a precedence diagram such as the one in Figure 2 that illustrates the hierarchical relation between the items presented in Table 1.

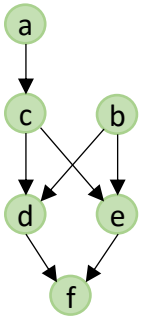


Figure 2 - Precedence Diagram for the Six Types of Algebra Problems presented in Table 1[3]

Table 1 - Six Types of Problems in Elementary Algebra[3]

Item	Name of problem type	Example of instance
(a)	Word problem on proportions (Type 1)	A car travels on the freeway at na average speed of 52 miles per hour. How many miles does it travel in 5 hours and 30 minutes?
(b)	Plotting a point in the coordinate plane	Using the pencil, mark the point at the coordinates (1, 3).
(c)	Multiplication of monomials	Perform the following multiplication: $4x^4y^4 \cdot 2x \cdot 5y^2$ and simplify your answer as much as possible.
(d)	Greatest common factor of two monomials	Find the greatest common factor of the expressions $14t^6$ and $4tu^5y^8$.
(e)	Graphing the line through a given point with a given slope	Graph the line with slope -7 passing through the point (-3, -2).
(f)	Writing the equation of the line through a given point and perpendicular to a given line	Write an equation for the line that passes through the point (-5, 3) and is perpendicular to the line $8x + 5y = 11$.

Looking at the precedence diagram in Figure 2, we conclude that we must master items *a, b, c, d* and *e* to get to item *f*, but there are six different options of moving down in the diagram. These are called the learning paths, and they represent different solution strategies that can be used to solve an item. For example, a student could learn the items starting in *a*, then *b, c, d, e* and *f*, or he could first learn *b*, then *a, c, e, d* and *f*. This is pedagogically relevant since students are not all similar, neither should be the teaching approach. Besides the different strategies represented by the learning paths, where the difference between them is the order they are learnt, depending on the knowledge structure characteristics, it might also be possible to master an item through different items. Thus, instead of

[3] Reprinted from "The assessment of knowledge, in theory and in practice", by Falmagne, J. C., Cosyn, E., Doignon, J. P., & Thiéry, N., 2006, *Lecture Notes in Computer Science, Vol. 3874 LNAI* (Issue 949)

having only AND nodes, we could have OR nodes. We will cover this topic in more detail in section 2.4 Closure Under Intersection.

Each knowledge state includes the items mastered by the student until that point. For the example illustrated in Figure 2, there are ten feasible knowledge states, and they are

$$K = \{\emptyset, a, b, ab, ac, abc, abcd, abce, abcde, abcdef\}$$

K is a set that represents a knowledge structure. It includes an empty state, where the student has no knowledge of the domain, and a state containing all the items in the domain, meaning the student has full knowledge on the subject.

2.2. KNOWLEDGE ASSESSMENT OUTPUT USING KST

When knowledge assessment is performed using a knowledge structure based on the Knowledge Space Theory, both the student and teacher receive one main output: the knowledge state where the student is.

It is of utmost importance to clearly define the boundaries of a knowledge state. Two basic concepts are crucial at this stage: the inner fringe and the outer fringe (Falmagne et al., 2006).

If the output was only a knowledge state, one would know what type of problems the student has mastered, but that is insufficient. The assessment result should guide the student, telling him/her what he/she is ready to learn next. The outer fringe gives us that exact list of items. Let us use the set K as an example. Imagine the student is in the state 'abc'. If we add one other item to this state and we get an existent state, then this extra item is part of the outer fringe of state 'abc'. States 'abcd' and 'abce' are feasible states, so items 'd' and 'e' would form the outer fringe of 'abc'. These are the items that the student should learn next to enlarge his/her knowledge.

On the other hand, we have the inner fringe. The list of items given by the inner fringe represent the items that should be monitored and reviewed because those are the most recent items learnt. The state 'abc' has two states immediately before, 'ab' and 'ac', so if we take 'b' from 'abc' or 'c' from 'abc' we get existent states; that means 'b', and 'c' make the inner fringe of 'abc'. These two lists are the answer to the questions "What is the student ready to learn?" and "What are the most advanced items the student has mastered?" respectively.

Instead of outputting the student's knowledge state along with all the items he has mastered, the combined outer and inner fringes provide both the student and teacher, in an informative and compact format, the result of the assessment and the information needed to proceed with the learning process.

Falmagne et al. (2006) state that experience in the school mathematics domain has shown the two lists – the outer and inner fringes - enclose on average 11 problems. The authors give an example to reinforce that the fringes make the output much more compact - from a knowledge state with 80 items, the two fringes contain only 9 items, which account for all the others.

2.3. KNOWLEDGE SPACES VS LEARNING SPACES

Until this point, knowledge structures were presented. The concept has essential ramifications that should also be discussed here: knowledge spaces and learning spaces.

A knowledge structure is a knowledge space if the two following conditions are met (Doignon & Falmagne, 1985).

(1) The empty set and the set with all the items in the space are knowledge states.

(2) It is closed under union, meaning that if we join two states, the result is also a feasible knowledge state of K.

If we consider a very large knowledge structure, (2) is a critical property since it enables a summary of a knowledge space by a minimal subcollection of its states, also known as the base of the knowledge space (Doignon & Falmagne, 2016). In this thesis domain, regardless of its relatively small size, this is a property to consider, as we might want to integrate the resulting knowledge structure with other knowledge structures, making the resulting space much bigger.

For a knowledge space to be considered a learning space, two additional conditions must be satisfied.

(3) Learning Smoothness – implying the items can be learnt one by one.

(4) Learning Consistency - implying that knowing more does not prevent learning something new.

To better understand this last condition *(4)* suppose a learner in a state M can master a new item q. Then, any learner in some state L including M either has already mastered q or is also capable of mastering it (Doignon & Falmagne, 2011).

Both these conditions make perfect sense in a pedagogical framework. Students should be able to learn one subject at a time, and regardless of what they learn first, they should always be given a chance to learn everything there is to be learnt.

2.4. CLOSURE UNDER INTERSECTION

In the previous section, we have seen why closure under union is an important characteristic of any knowledge space, but what about closure under intersection? What does it imply?

The formal concept tells us that if we intersect two states of a ∩ - closed knowledge space, the result is also a feasible knowledge state of K.

In practice, what it means is that each item in the space has a unique set of prerequisites (Doignon & Falmagne, 1985). Pedagogically speaking, this assumption makes no sense as there can be several strategies for mastering a specific item and consequently, the antecedents may vary according to the adopted strategy (Doignon & Falmagne, 2016).

Falmagne et al. (1990) built the knowledge space K_1 for their five problems'[4] domain $\{1, 2, 3, 4, 5\}$. The five problems can be seen in the table below.

[4] In this example, the knowledge structure is not made of items, but of instances of the items. See more about this in section 2.6.

Table 2 - Five Problems in High School Mathematics[5]

Problem ID	Problem
1	$378 \times 605 = \ ?$
2	$58.7 \times 0.94 = \ ?$
3	$\frac{1}{2} \times \frac{5}{6} = \ ?$
4	$What\ is\ 30\%\ of\ 34\ ?$
5	$Gwendolyn\ is\ \frac{3}{4}\ as\ old\ as\ Rebecca.$ $Rebecca\ is\ \frac{2}{5}\ as\ old\ as\ Edwin.$ $Edwin\ is\ 20\ years\ old.$ $How\ old\ is\ Gwendolyn?$

$$K_1 = \{\emptyset, \{1\}, \{3\}, \{1,2\}, \{1,3\}, \{1,2,3\}, \{1,2,4\}, \{1,2,3,4\}, \{1,2,3,5\}, \{1,2,3,4,5\}\}$$

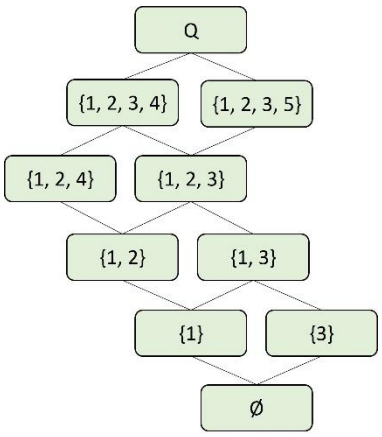


Figure 3 - Graph of the Knowledge Space K_1

[5] Adapted from "Knowledge Spaces", by Doignon, J. P., & Falmagne, J. C., 1999, *Springer-Verlag Berlin Heidelberg*.

However, the authors rejected K_1 as it was the result of a model which did not allow an item to be mastered using different strategies. They decided to use as a working hypothesis the knowledge space K_2, which was not closed under intersection and consequently made more pedagogical sense.

$$K_2 = \{\emptyset, \{1\}, \{3\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{3,4\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\},$$

$$\{1,2,3,4\}, \{1,2,3,5\}, \{1,2,3,4,5\}\}$$

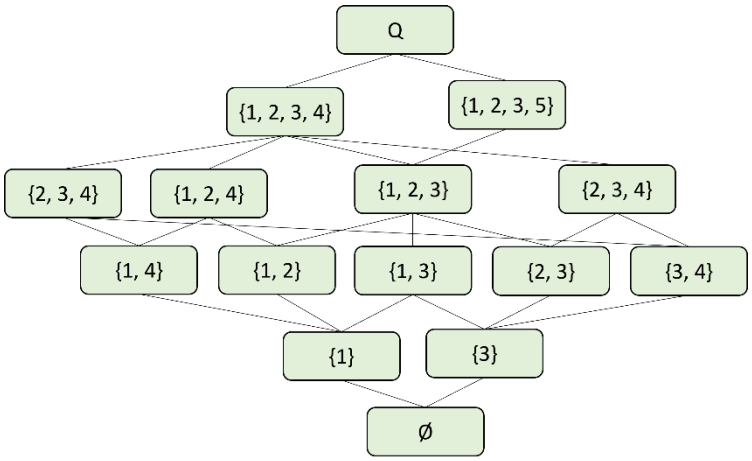


Figure 4 - Graph of the Knowledge Space K_2

Considering K_2, imagine a hypothetical case of a student who has solved problem 4. What inferences can be drawn from this? The student's current knowledge state must have problem 4, but the only reasonable inference that can be made is that his state also contains problem 1 or problem 3. At least one of these problems must be part of his/her current state, but it is not possible to know which one. There are two different ways of mastering problem 4.

As mentioned above, problem 4 cannot be solved without mastering either problem 1 or problem 3. In other words, state $\{4\}$ must not be a feasible knowledge state. If K_2 was closed under intersection, this state would have to be a feasible knowledge state as it results from the intersection between two feasible states - $\{1,4\}$ and $\{3,4\}$.

In an ∩ - closed knowledge space, items are connected to their prerequisites only by and-nodes, while in a not ∩ - closed knowledge space, the connection can also be made through or-nodes.

Another practical example would be the one given by Desmarais et al. (2006) – a student might learn to add two fractions by transforming them into a common denominator, while someone else might have learned to turn them into decimal form first and then back into a rational form.

2.5. How To Find The Items and Instances Of a Domain

To find the items in the domain of beginning algebra, Doble et al. (2006) examined a sample of textbooks about the subject and got between two and three hundred items. Regarding instances, textbooks can also be a good tool, as well as previous tests made to students about the subject under study.

The number of topics and items highly depend on the domain. ALEKS (ALEKS Corporation, 2013) is one of the most widely known adaptive learning programs, and the foundation of their work is the Knowledge Space Theory. ALEKS assesses students' knowledge, determining their ability level and delivers insightful progress reports to the professors (Scalise et al., 2007). In the vast domain of Higher Education Math, ALEKS' team encountered 11 topics, each one with 4 up to 11 sub-topics and each sub-topic had between 1 and 13 items, accounting for 314 items (*Higher Education Math Placement*, 2016).

Both ALEKS (ALEKS Corporation, 2013) and Doble et al. (2006) have a base of instances that are populated with randomly selected values when the instance is chosen in a test. This makes cheating between colleagues much more difficult because it is almost impossible for two students sitting next to each other to get the same values. With just some instances, this strategy increases the number of questions available exponentially.

Until now, we have talked about items, but another concept can be useful when discovering and creating the items in the domain. Besides the content to be learnt, which can be thought of as the items, learning-centred approaches go deeper and account for learning objectives. These represent what the learner will be able to do if he/she can successfully solve an instruction/problem (Marte et al., 2008). Learning objectives can be expressed through action verbs, such as "*state*" or "*apply*" (Albert et al., 2005). If learning objectives are added to the items found, we will be able to know more about what the student really knows.

Some frameworks have been developed to ease the usage of learning objectives in teaching. Using Anderson et al.'s (2001) words: "All frameworks such as the Taxonomy are abstractions of reality that simplify in order to facilitate perceptions of underlying orderliness.".

The verbs used in a learning object refer to the intended cognitive process. Marte et al. (2008) reviewed the work done by Anderson et al.'s (2001) and they came up with 6 categories representing the cognitive process dimension. From the first category to the last, we have an increase in cognitive complexity. It is important to note that not all combinations of items and verbs may be meaningful (Albert et al., 2005).

Table 3 - The Six Categories of Cognitive Processes[6]

Process Categories	Description and Examples
Remember	Retrieve relevant knowledge from long-term memory, e.g., reorganise dates of important events in US history
Understand	Construct meaning from instructional messages, including oral, written, graphical communication, e.g., classify mental disorders, make a summary, draw a logical conclusion from presented information
Apply	Carry out or use a procedure in a given situation, e.g., divide one whole number by another whole number
Analyse	Break material down into constituent parts and determine how parts relate to one another and to an overall structure, e.g., differentiate between (ir)relevant numbers in a word problem
Evaluate	Make judgements based on criteria and standards, e.g., judge which of two methods is the best way to solve a given problem
Create	Put elements together to form a coherent or functional whole; reorganise elements into a new pattern or structure, e.g., plan a research paper on a given historical topic

2.6. How To Go From The Individual Nodes To a Knowledge Structure

Once the items (and instances) in the domain are identified, some known techniques are used to uncover the surmise relations on the sets of items (M. T. Segedinac et al., 2018).

The concept of surmise relation was introduced by Doignon & Falmagne (1985), and it represents a prerequisite relationship between items within a body of information for the assessment of knowledge. These relationships make it possible to infer which sets are knowledge states and which are not, reaching a knowledge structure. All knowledge states are subsets of items, but not all subsets are knowledge states.

There are different approaches to constructing a knowledge space. Let us elaborate on four of them. The first, introduced by Koppen & Doignon (1990), is based on a systematic questioning of experts.

The second approach is based on students' response patterns and data analytics techniques. Multiple methods have been developed to perform this analysis. The resulting space is sometimes called the real knowledge space (M. Segedinac et al., 2010) because it is not based on what the experts expect it to be but on real data from the students.

The third approach was proposed by M. Segedinac et al. (2010), and it uses the analysis of problem-solving processes. This approach does not need previous student data, but it does involve experts. However, the amount of work asked from them is much lower than in the procedure proposed by Koppen & Doignon (1990).

[6] Adapted from "Activity- and taxonomy-based knowledge representation framework", by Marte, B., Steiner, C. M., Heller, J., & Albert, D., 2008, *International Journal of Knowledge and Learning, 4(2–3), 189–202.*

There is an important difference between the first approach and the last two approaches. Until this point, it has been implicit that the nodes in the knowledge space are the items, but this is not always true. In the first approach, the nodes of the knowledge space can be the items or the instances of the items, but in the last two approaches, the nodes of the knowledge space must be the instances and not the items.

Unlike the nodes, the edges always represent precedence relations, but these relations can have different meanings. In the third approach, the meaning of the edges differs a little from what has been presented so far. Here, the precedence relations represent the complexity of the problems instead of the content knowledge needed to solve the problems. The ability to solve less cognitively complex problems is taken as a precondition for solving more cognitively complex ones (M. Segedinac et al., 2010).

The fourth approach under consideration accounts both for the instances of the items and the items themselves. It is based on an extended version of the Knowledge Space Theory called Competence-Based Knowledge Space Theory (CBKST), introduced and put to practice by Ley et al. (2010). It does not need as many hours from the experts as the querying the experts' approach, and it delivers two models - the User Model and the Domain Model – which will be explained in Section 2.6.4.

The following subsections present with more detail each of these approaches to construct a knowledge space.

2.6.1. Querying Experts

The knowledge space obtained through this approach is built according to experts in the domain of knowledge. They are asked a series of questions about the items (or instances) of the domain and their precedence. Based on their answers, the knowledge space is constructed. The nodes of the resulting knowledge space can be the items or the instances of the items, which is convenient because depending on the what is desired, one can be more suitable than the other.

Having the instances as the nodes can be seen as having the highest level of granularity possible. This can be particularly convenient if there is a closed set of instances to which we want to discover the inherent knowledge structure. However, when more instances are expected to be added to the equation, this does not seem to be the best option as they would represent new nodes and it would be necessary to recalculate the entire network each time an instance is added. As the items are expected to be more static, having them as the nodes will result in a more high-level but stable structure. Also, one important consideration to make regards the storage capacity. When the instances are the nodes, the number of nodes will be much higher.

The main advantage of the expert querying method lies in the fact that it allows the construction of a knowledge space without having any previous data on students' knowledge. The most serious flaw of this method is closely related to its main advantage. Not having students' data can make it easier to reach a first knowledge space, but it will be what the experts expect it to be.

Additionally, this approach requires many hours of work from the experts. Baumunk & Dowling (1997) consider that multiple experts with different backgrounds and levels of experience should be questioned in order to confront different views, which means many hours from each of the experts. Different experts have different responses, which creates another issue to be addressed when using this approach that lies in the need to gather the information given by all the experts. Baumunk &

Dowling (1997) integrated all the knowledge spaces into one, only considering the surmise relations on which all the experts agreed. This was called a consensus space. A majority space was also built taking into account the surmise relations where the vast majority of experts agreed.

Procedures have been created to ease the questioning process. The most widely known is the QUERY procedure, described by Koppen & Doignon (1990) and by many others over the years (Kambouri et al., 1994; Koppen, 1993). It only uses questions of type **[Question]**.

> **[Question]** "Suppose that a student has just failed all the problems a_1, a_2, …, a_k. Is it then practically certain that this student will also fail problem b?"

The subset $\{a_1, a_2, \ldots, a_k\}$ can be named A, and the above question can be interpreted as follows:

> **[Interpretation]** "Is it true that in the knowledge space, there is no state containing item *b*, but not the subset A?"

Even considering questions like **[Question]**, when the domain contains m items, the number of questions would be $m \times 2^{m-1}$. In a very small domain with 15 items, there would be 245 760 questions. Fortunately, the questions in **[Question]** are not all independent, and inferences can be drawn from previously asked questions, reducing the collection of questions to be asked.

Computationally, considering all the possible questions at the beginning of the questioning is a burdensome task, and Koppen & Doignon (1990) proposed a procedure that divides the questioning into blocks. The set of feasible knowledge states and the questions for block *n+1* are only computed after block *n* is closed, leading to fewer questions in each block. Block 1 contains one-to-one questions, where set A has a single item, in block 2 the set A has 2 items and so on. If the questioning stops after the first block, the knowledge structure achieved is closed under union and intersection. However, if the process continues until no more questions are left – the next block is empty - the final knowledge space will no longer be closed under intersection, which makes more pedagogical sense but costs much time to experts.

Baumunk & Dowling (1997) used this querying strategy, and they did not query the experts on premises with a size of A longer than one due to the experts limited time. Koppen (1993) refers to an implementation of this procedure done by Kambouri et al. in 1991 with a domain of 50 problems and five experts. Four of the five experts terminated their task after five blocks of questions, and depending on the expert, the task took 10 to 20 hours (not including breaks), and each expert responded between 1000 and 2500 questions, a lot less than the initial $50 * 2^{49}$. The other expert stopped after the third block because as she had a more conservative approach (more negative answers), there were too many questions still to be asked. The advantage of the procedure is that even if it is interrupted before the end, it guarantees that vital states are not lost (Koppen & Doignon, 1990). This means that the knowledge space obtained includes the one which would have been produced if the process had ended regularly.

This approach looks promising when previous data on the students does not exist. However, the fact that it would take many hours from the experts, alongside the lack of an already available algorithm for experts questioning, are two limitations for its implementation. This algorithm would have to: 1) account for all the possible inferences; 2) keep track of all the answers already given; 3) confront them with the new answer in each iteration; 4) look for the best question to be asked next – considering the

number of possible inferences to be taken from it (Koppen, 1993) – and; 5) compute the blocks of questions in each phase.

Many journal articles describe the functioning of such an algorithm, but no implementation is available. In Kambouri et al. (1994), it is only said that the QUERY procedure was programmed in Pascal, and the user interface was written in C. Dowling et al. (1996) published a journal article introducing a graphical user interface integrated into a procedure for acquiring knowledge from the experts using C and a C++ library and at that time it was made accessible, but it is no longer. Creating such an algorithm, making it available to the scientific community and providing a detailed description of its implementation would be of great interest to future work in this area of study.

2.6.2. Data Analytics in Students' Responses

Data-analytic techniques look for patterns in students' responses and identify the surmise relations on the sets of questions.

These techniques need no involvement of experts because of the exclusive use of data from students' knowledge to build the knowledge space. M. T. Segedinac et al. (2018) name this the real knowledge space. However, it is not sure that the real knowledge space is found, at least not for the entire population. Over-fitting may occur depending on the sample of students chosen to answer the questions (Desmarais et al., 2006).

The nodes are the problems themselves, also called tasks, and a test containing those tasks is presented to the students for later analysis using data analytics techniques.

M. T. Segedinac et al. (2018) used the Corrected and Minimized Inductive Item Tree Analysis algorithm (IITA), available in R programming language[7] (Ünlü & Sargin, 2010), to identify the surmise relations in the set of tasks. It is used for deriving implications from dichotomous data (Ünlü & Sargin, 2010), so the students' answers must be considered either corrected (1) or incorrect (0). Using the IITA, some tasks can be considered outliers if no patterns regarding these tasks are identified in the data.

Many other techniques have been developed to discover knowledge structures from students responses. Doroudi et al. (2016) introduced Sequencing Constraint Violation Analysis (SCOVA), a method for evaluating alternative activity sequences using existing data. De Chiusole et al. (2017) created an extension of the *k-modes* algorithm called *k-states,* which extracts a knowledge structure from students' data. Desmarais et al. (2006) suggested two approaches to identify item to item structures from empirical data: Bayesian Network structural learning and Partial Order Knowledge Structure induction algorithm.

Due to the unavailability of students' data, the construction of the knowledge space using this approach will not be possible, and a deeper analysis of the literature available on this subject is out of scope. If this limitation did not exist, this would be a path to explore as it barely depends on experts to build the structure, and it tends to be more realistic than the ones using experts' knowledge.

2.6.3. Analysis of Problem-Solving Processes

This approach divides the problems to be solved by students into sets of sub-problems (Segedinac et al., 2018), and its core is the complexity of the problems. The task complexity will not entirely predict

[7] https://search.r-project.org/CRAN/refmans/DAKS/html/iita.html

the student performance, but it is a proxy as it should correlate with it (Knaus et al., 2011). A numerical complexity value is assigned to each task/instance, providing the teachers with a greater awareness of the relative complexity of the topics assessed.

Its main advantages are the fact that it can be applied without prior information on students' knowledge and the reduced involvement of experts in the process when compared to the querying the experts' approach. This is a good approach in cases where there are few dependencies between the tasks in terms of content, because the relations between tasks will not be based directly on the content of the tasks but on their complexity. Even if tasks are independent, the complexity of each task will most likely differ and as the structure will be built based on tasks' complexity, there will be a hierarchy based on complexity and the structure will not be too flat, allowing a proper knowledge assessment. The adequacy of this method to the work of this thesis will be discussed in the following paragraphs.
This method's main advantage can also become its most significant disadvantage. As data on students' knowledge is not considered, the knowledge space will be the space the experts expect it to be.

An important note on the work developed by Segedinac et al. (2018) is that it results in knowledge spaces closed both under union and intersection.

The complexity of the problems is the core of this approach, and it is the complexity that gives meaning to the edges of the knowledge space, so it is important to understand its relation to the problems' difficulty felt by the students. Lee & Heyworth (2000) presented the cognitive difficulty of a problem as complexity and concluded that the students' perceptions of complexity roughly correspond to the cognitive difficulty of the problem. Robinson (2001) concluded that cognitive complexity significantly impacts the perception that students have of task difficulty – when complexity increases, stress increases, while interest and motivation remain the same. Therefore, a knowledge space based on problem complexity can be valuable to map the students' minds.

However, when debating on the purpose of the thesis – building the knowledge space for an informative knowledge assessment tool - and implementing this approach, even if it delivers a result valuable to map students' minds, it does not fit the objective. The main goal of the assessment is to track the students' progress in the domain's content and provide to the student and/or teacher the content that should be learnt next. However, if the precedence is related to the complexity of the tasks the only possible conclusion is to point the student to the next problem in the complexity hierarchy. The fact is that the space will not be built based on the content – items - of the domain, but on the complexity of each question and this is not what is pretended.

2.6.4. Competence-Based KST approach

Competence-Based Knowledge Space Theory overcomes the KST limitation of being a purely behaviouristic approach (Albert et al., 2005). It considers the student performance, seen as the observable solution behaviour of a learner on the set of problems in the domain, and his/her competencies (skills), conceived as a latent, more cognitive level explaining the performance (Albert et al., 2013; Korossy, 1999).

Ley et al. (2010) suggested a methodology to construct the models for an adaptive learning system using the theory of CBKST.

The methodology proposed includes the construction of two models, the Domain model - achieved through the collection and structuring of experts' knowledge, resulting in a performance structure -, and the User model – modelling the learner's available knowledge through competency states. Accounting for both models, this approach seeks to derive a student's learning needs from the task he/she will perform next and their current knowledge.

While in KST, the result is a knowledge structure, in CBKST, there is a performance structure and a competence structure. However, the concepts underlying these structures are the same as the ones introduced for a knowledge structure.

To build these models, two types of information are requested from the experts. A set of the typical tasks in the domain must be identified, as well as the competencies/skills required to solve those tasks successfully. After collecting this information, the succeeding stage is associating each task with the skills required to answer it.
Just like in the querying the experts' approach, one of the biggest challenges in this method is the process of collecting knowledge from the experts and structuring it. The method applied to simplify the collection of knowledge from the experts was the creation of a task-competency-matrix.
After identifying the typical tasks of the domain and the set of competencies needed to solve them, the experts used this matrix to assign to each task the competencies required to solve it successfully. Each minimal set of competencies capable of solving a task is a feasible competency state.

One of the characteristics of the final competence space is its closure under union (see (2) in page 7), which avoids the need of storing the entire competence space. One can only store its base: the competence states, which cannot be obtained by the union of two or more states. Then, the entire space can be obtained by closing the base states under union.

After having the competence structure, the performance structure is generated by assigning to each competency state the set of tasks that can be solved with the competencies in that state. Each of these sets of tasks is a feasible performance state, and each performance state is associated to a competence state. The current competence state of a student portrays the set of skills the student showed mastery. In other words, the student successfully solved a set of tasks comprising those competencies.

For a better understanding of the process of creating the competence and performance structures, a practical example will be given in the following paragraphs.

Consider a very small domain where only three competencies {*a*, *b*, *c*} and four tasks {*1*, *2*, *3*, *4*} were identified. The expert proceeded to the assignment of the competencies to the tasks and got the following task-competency matrix.

Table 4 - Example of CBKST Task-Competency Matrix

TaskID/CompetencyID	a	b	c	Minimal subset of competencies
1	X			{a}
2		X		{b}
3	X	X		{a, b}
4	X	X	X	{a, b, c}

From Table 4, tasks *1* and *2* require only one competency, while solving tasks *3* and *4* requires the student to master two and three competencies, respectively.
According to the identified tasks, competency *c* is only learnable after mastering competencies *a* and *b*, whereas *a* and *b* can be learnt independently. These inferences can be made only by looking at the matrix because this is a very small domain.

After completing the matrix, the precedencies between the domain competencies are inferred using a rule introduced by Ley et al. (2010):

[Precedence Rule]: *A prerequisite relationship is always assumed between two competencies when only one is present when the other is also present.*

In other words, skill *x* is an antecedent of skill *y* if every time *y* is used *x* is also used. If *x* and *y* are always together – there is no subset of skills containing only *x* or only *y* -, these are not considered prerequisites of one another, and the combination of these two skills in one single skill should be considered.
This rule can be implemented in a programming language like *Python*[8].

The competence and performance structures resulting from the inferences made using the matrix in Table 4 and the **[Precedence Rule]** are displayed in the figure below.

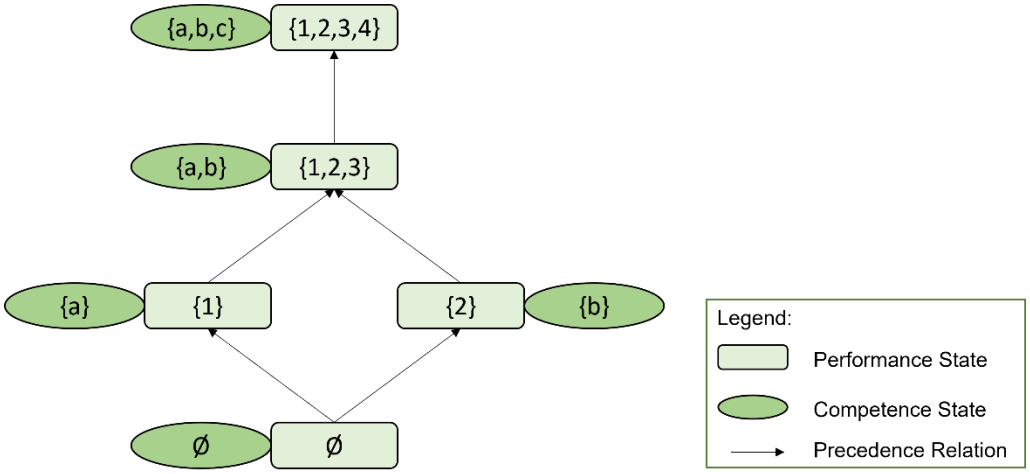


Figure 5 - Example of CBKST Competence and Performance Structures

The competence and performance structures could be presented separately, but the purpose of showing them together is to clarify the existing association between the two structures. As seen in Figure 5, each performance state contains the tasks which can be solved using the competencies in the associated competence state.

The performance structure on its own can be seen as a knowledge structure that is used to assess students' knowledge directly. The student is asked some of the tasks in the performance structure and based on the responses, a performance state is assigned to the student. The competence structure adds the skills to the equation. Instead of outputting the next task the student is ready to learn, with the competence structure, the main output will be the skills to be learnt next. This method delivers what is called a task-based competency assessment.

In practice, this is what happens in the assessment: the student answers some tasks; the current performance state (M) is assigned based on the responses given by the student; knowing M, the associated competence state (N) is also known; compute the difference between the skills in N and the skills in state N+1 (N+1 is a consequent state of N, which ideally has one more skill than N); the additional skills are the ones the student should learn next.

The choice of the approach to follow from the four approaches presented will be made in chapter 3.1 - Methods Comparison and Discussion.

[8] https://www.python.org/

2.7. Types of Questions in Knowledge Assessments

The decision on the types of questions to perform in a students' knowledge assessment plays an important role to minimize the hypothesis of lucky guesses and careless errors.

Hamtini et al. (2015) built an intelligent educational assessment tool. The quizzes made to the students had questions in four formats: fill in the blank, multiple-choice, multiple choice with more than one correct answer and true or false. They offered the teacher the possibility of including pictures in the questions, and students could also answer with pictures, which might be a good strategy for more complex questions if visual recognition is integrated into the tool.

Regarding the formats used by Hamtini et al., mainly the multiple-choice and true or false, these are not the most reliable due to the high probability of lucky guesses. This issue is problematic, and strategies to fight it are imperative because it can endanger the reliability of the tool, assuming the student has mastered some item, that in fact, he has not.

The probability of lucky guesses is even higher in True/False than in Multiple Choice, as the possible options are just two, and consequently, there is a 50% chance of picking the correct answer. One way of discouraging lucky guesses is taking points from the final grade when an incorrect response is provided.

In their application, Doble et al. (2006) opted to use open responses and multiple-choice with a large number of possible responses as a way to minimize lucky guesses. Obviously, as the number of options increases, the least are the chances of a lucky guess, but it is not viable to have too many options. The challenges of having too many options do not only concern the learners who will answer them but also the teachers who will create those options. Reading and answering multiple choices with many options makes the process tiresome, and creating the options is also challenging. The more options, the more difficult it gets to create plausible ones (Vyas & Supe, 2008).

Research suggests that the optimal number of choices is three (Raymond et al., 2019; Vegada et al., 2016), but some studies report no difference between the reliability of three and four options multiple choice questions (Dehnad et al., 2014).

When comparing open-ended questions to multiple choices, Kang et al. (2007) found that open-ended questions with feedback can improve student learning more than multiple-choice questions.

The platform ALEKS has grown over the past years and today is one of the most effective adaptive learning programs. ALEKS assessments aim to check students' knowledge accurately, so they use no multiple-choice questions, only open-ended problems that require the student to understand, learn and master the topic. To allow this, one of the focuses of ALEKS developers is their Answer Editor. The input tool must be easy to use and must be aligned with the topics. The input is always in the form of proper expressions and constructions (ALEKS Corporation, 2013).

Concluding, multiple-choice and true or false questions would be much easier to implement. However, open questions are more reliable and improve learning in a way the other two options cannot. This results in a trade-off between the tool reliability and its complexity.

ALEKS, in line with the research of Falmagne et al. (2006), always provide in its questions the option "I do not know". This option may not seem a big deal, but it helps a lot in assessing student knowledge and plays a big role in its reliability. When a student gives an incorrect answer, there is always the possibility of it being the result of a careless error. However, if he/she chooses the "I do not know" option, then it is certain that the student has not mastered the items needed to solve it.

This research will be considered in due course when deciding on the type of questions to ask.

2.8. How to Visualize and Store a Knowledge Space

This thesis aims to create a knowledge space in the domain under study. However, it is essential to store and visualize the resulting structure to conclude about the space validity and possible improvements.

Considering that a knowledge structure is made of nodes and relations, a relational database (RDBMS) is not the most adequate as it is not meant to store highly connected data. The words *nodes* and *relations* mentioned above to refer to the items of a knowledge structure and their precedencies, already point in the direction of a specific type of database – graph databases.

Graph databases are directed acyclic graphs (DAGs), useful structures for modelling objects and interactions (Vicknair et al., 2010). They can be thought of as a flowchart linking causes and effects (Foraita et al., 2014). In the case at hand, the causes would be the antecedents and the effects would be the precedents. The edges being directed, means they point in a specific direction. Also, when a node has no associations, it is left unconnected (Shrier & Platt, 2008). The acyclic keyword indicates that for any given node, there is no edge leaving from it that leads to a path that comes back to that same node.

These characteristics of graph databases seem to be adherent to the characteristics of the data at hand. This kind of database has been used to represent, among other things, website link structures and social networks.

Comparing to RDBMS, graph databases are much more flexible when it comes to adding new data or data relationships to the model, and they enable the inclusion of complex data and dependencies (Lazarska & Siedlecka-Lamch, 2019). Concluding about graph databases, they seem like a good fit for the task at hand.

Besides storing the knowledge structure, it is important to visualize it. Gephi[9] does both. It is an open-source platform for visualizing and manipulating large graphs and can visualize any knowledge structure. To build a network graph, Gephi stores the information on the nodes and edges.

There are some intelligent tutoring systems, like ALEKS and Khan Academy[10], that allow the navigation through their existing knowledge structures and their storage. However, they did it as an integrated part of their system, and it is not possible to integrate the framework used with other tools besides

[9] https://gephi.org/
[10] https://www.khanacademy.org/

their own. Nasar (2016) noticed this need and developed a knowledge space data-store, using graph databases and an open framework API on top of it. The graph database chosen by Nasar was Neo4J[11].

Another possibility for storing and visualizing a knowledge structure is a package available in R programming language, named *kst*. It allows the storage of a knowledge structure in a variable by passing to the function *kstructure()* the states of the structure. Afterwards, it is possible to plot the structure using a directed graph, also called Hasse diagram (Kickmeier-Rust & Albert, 2015), by calling the *plot()* method. This package allows inclusively to validate if a knowledge structure is a knowledge space, which is a valuable insight as pedagogically speaking a structure used for a knowledge assessment should be a knowledge space (Stahl & Hockemeyer, 2019).

There is an alternative to the referred *kst* package, called *DAKS,* that has some functions which might be of interest (Ünlü & Sargin, 2010). The method *hasse()* is similar to *plot()* in the *kst* package, and there is a function that transforms a set of knowledge states to the corresponding set of surmise relations and a function to do the opposite is also available.

[11] https://neo4j.com/

3. METHODOLOGY

First, a reminder on the domain under study is of importance. The domain goes from the most basic SQL knowledge to operations like Deleting, Inserting and Updating records in the database, and it refers to the SQL programming language in a Microsoft SQL Server context (Transact SQL).

3.1. METHODS COMPARISON AND DISCUSSION

Reviewing the four approaches introduced in 2.6 that start from individual nodes and create a knowledge structure, CBKST seems to be the most adequate given the advantages and challenges of each method and the time and resources available. The data analytics method requires students' data, and the problem-solving approach was not a good fit for the present work, leaving two alternatives: the experts' querying and the CBKST. CBKST was chosen to create the knowledge structure of the domain under study due to the implication of less effort from the experts, a less subjective outcome and the combination of skills and tasks it provides. For the sake of simplicity, the experts' querying approach will be addressed as A_1 and CBKST as A_2.

One could argue why A_2 is considered to be less subjective than A_1.
In A_1, the relations displayed in the structure directly result from the experts' answers, whereas in A_2, the relations are derived from the set of identified tasks. These tasks are identified by the experts, and the experts are asked to assign competencies to them, but the result will not be so biased on direct precedencies assigned by the experts as it happens in A_1.

When teaching, one of the efforts made by the professors is to create a set of tasks to assess the students' knowledge along the way of the course. Theoretically, the students should get to the end of the course knowing how to solve each of those tasks. The CBKST method, based on those tasks and the competencies addressed, delivers a structure that guides the learning process so that, at the end, all the tasks (and respective competencies) are mastered. The outcome of A_2 is highly dependent on the set of identified tasks, while the result of A_1 is highly dependent on the experts and their subjective domain knowledge.

As a result of the chosen method, the outcome of this work will not be one knowledge structure but two – a competence and a performance structure – which combined deliver a strong basis for an assessment tool. The CBKST method will be the starting point, but some add-ons will be made to enrich the final solution.

Even though the CBKST approach proposed by Ley et al. (2010) seems to be the right choice, some questions arise when debating on implementing this method in the domain under study. In section 233.2, these questions will be discussed.

If the **[Precedence Rule]** is to be applied, the set of tasks must be complete; otherwise invalid precedencies can be easily inferred. Those precedencies may be real in the sample of tasks considered but not in the domain. As in most data mining problems, the biggest the dataset, the more realistic and robust is the result obtained. In this thesis, to create a dataset as complete as possible, CBKST method will be improved by turning it into an iterative process. The diagram below displays the stages to be implemented.

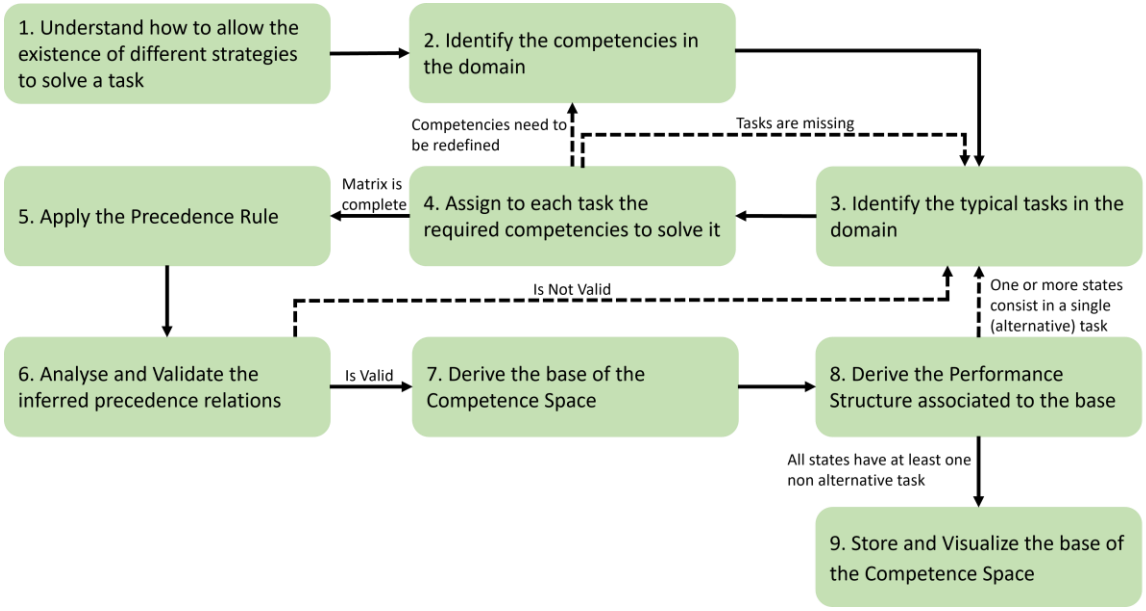


Figure 6 - Methodology Diagram – Steps to Follow

In Figure 6, the continuous arrows represent the natural flow of the process and the dashed arrows illustrate the stages where a step back may be needed.

Additionally to the iterative nature of this process, stage 6. is also an add-on to the initial method. By validating the precedencies derived by the precedence rule, it will be possible to detect invalid inferences and make the set of tasks more complete.

An intermediate step, "Validate the structures with students' data", between stages 8. and 9. would be of importance to confront the expected competence space – built using experts' knowledge and experience - with the "real" competence space (M. Segedinac et al., 2010) – built from students' response patterns. Due to the time and resources available, it is considered as future work. However, the implementation of the other stages will account for the existence of this extra one. Namely, before identifying the typical tasks in the domain, a potential platform to be used in the students' questioning will be considered to avoid being creating tasks that are not feasible in the chosen platform.

The methodology will roughly follow the structure displayed in Figure 6 - Methodology Diagram, based on the adapted CBKST approach discussed above, and each of the stages will be discussed on its own time, as well as the cycles present in the diagram.

Multiple authors (Baumunk & Dowling, 1997; Koppen, 1993; M. T. Segedinac et al., 2018) support the inclusion of more than one expert during the process of collecting information to reduce the inherent subjectivity of the approaches highly dependent on the experts' knowledge. For the competencies and typical task identification and the validation of the competence structure resultant from the **[Precedence Rule]** the author of this work, with two years of teaching experience in the domain, was assisted by an expert in the domain, with ten years of teaching experience in this area of study.

3.2. How to Allow the Existence of Different Strategies to Solve a Task

Going back to debating the implementation issues of the chosen method, in the SQL programming language, there is not a single way to answer a question (i.e. if, in a query, a person wants to return the records where the variable X has the value $'Radish'$, this can be achieved by using in the *Where* clause the expression $X = 'Radish'$ or the expression $X\ IN\ ('Radish')$. There can be different strategies, or in other words, alternative minimal subsets of skills to solve a task.

The existence of different strategies to master a task is related to the assumption of closure under intersection (see 2.4), which results in the existence of a unique strategy to master a task.
Thinking on the **[Precedence Rule]**, the competency *x* is an antecedent of *y* if it is present in all subsets including *y*, so if *x* is missing from one (or more) subset(s) it no longer is an antecedent, however, that might have happened because an alternative strategy was used. This rule will not capture this and both *x* and the alternative skill(s) used will not be connected to *y*. This looks like a flaw but a deeper analysis is necessary.

To facilitate the understanding of the following paragraphs, it is important to recall the composition of each structure and its association. The objects of the competence structure are the competencies, and the objects of the performance structure are the tasks. Each competence state has an associated performance state composed of the tasks which can be solved by the competencies in the respective competence state.

Having the precedence rule mentioned above, is it possible, in the CBKST method proposed, to guarantee the possibility of solving a task using different strategies? Ley et al. (2010) do not address this issue, but in the domain under study, it is essential to consider this possibility. The solution proposed consists of considering each alternative subset of skills capable of solving a task as an independent task in the matrix.

In the stage of filling the task-competency matrix, by definition (Ley et al., 2010), each column represents a skill and each row represents a task, but in the matrix to be created, each different strategy will also be seen as a row. *Task1_alternative1* and *task1_alternative2* would identify two rows of the matrix. Table 5 represents a possible task-competency matrix where Task1 can be solved through two alternative strategies.

Table 5 - Example of CBKST Task-Competency Matrix with Alternative Strategies

TaskID/CompetencyID	a	b	c	Minimal subset of competencies
Task1_alternative1	X			{a}
Task1_alternative2		X		{b}
Task2		X		{b}
Task3	X	X		{a, b}
Task4	X	X	X	{a, b, c}

Having the alternative strategies as separate rows will positively impact on the competence structure, as the number of subsets of skills used to inference the precedencies will be higher and consequently more robust, minimizing the chances of encountering invalid precedencies.

The impact of having alternative subsets of skills to solve a single task is not as promising in the performance structure because each alternative will be seen in the structure as a different task, but it

is unpracticable to ask the student the same question more than once. Nevertheless, there is a way to overcome this obstacle.
Consider the following two items from the set of tasks: *task1_alternative1* and *task1_alternative2*. These represent the two strategies that can be used to solve task one, and they are seen as independent tasks. Assume a student has used *task1_alternative1* to solve task one.

The solution to guarantee that a student is not asked the same question twice is to exclude from the performance structure the alternatives which were not used. In the hypothetical case mentioned, *task1_alternative2* would be excluded from the performance structure and consequently task 1 would not be asked again. However, what if *task1_alternative2* was the only task in a certain state? This would mean that a state (in both structures) would become unreachable, as there would be no other task to test the student's knowledge in that specific set of skills (competence state). To make sure this does not happen, one must make sure that the states which contain alternative strategies also contain other tasks (i.e., the performance state containing *task1_alternative2* would at least contain one other item that could be *task6*_alternative1).
With this planning, no state will become unreachable due to the inexistence of more tasks unless all the items in it are excluded.

As there are different strategies to solve a task, a student may constantly prefer the same strategy over a different one, resulting in the exclusion of all items in a certain performance state. It would be incorrect to assume that the student has not mastered the items in the excluded state. For example, a student might always find alternatives to the skill *"Find column values belonging to an explicit set (IN)"*, not because she/he does not master this skill, but because she/he prefers to use the alternatives.
The solution when a student is constantly "escaping" a certain skill can be to present along with the task, the skill(s) the learner must put in practice, directing the answer to use the skills missing in his/her portfolio. If the student answers the task correctly but still does not use the suggested skill(s), the algorithm can conclude he/she has not mastered those skill(s). If the student uses it correctly, the assessment continues.

Moving forward, the domain under study is characterised by the impossibility of knowing *a-priori* all the possible minimal subsets of skills capable of solving a task, however a great advantage of using the task-competency matrix is the easiness in adding or removing tasks or competencies from the structures, which mitigates the negative impact of this characteristic of the domain and allows the structures not to be static.
When a new minimal subset of skills is found it can be easily added to the structures, one just must be careful not to be creating a new state that will only contain that one alternative strategy (i.e., *task1_alternative3*). This would lead to the possibility of having an unreachable state, as mentioned before.

To avoid this, the addition of a new strategy to solve a task should always be followed by adding a different task with the same subset of skills. In fact, more tasks in each performance state would be better as it increases the quality of the knowledge structure.
These paired additions guarantee that when an alternative strategy is excluded because a different strategy was used, there are more tasks in that same performance state, which allows the student to have another try to show that he/she has mastered the skills in the associated competence state.

A clarification on the term "alternative subsets of skills" is of importance. It has been explained how to consider the existence of different strategies to solve a task, but one cannot confuse these different strategies with the location of these strategies in the structures. *Task1_alternative1* and *task1_alternative2* are alternative strategies but that does not imply that they are close to each other in the performance structure.

Let's look into one practical example. Consider the task below.

> **[Task1]**: *Write a query that returns the CrimeID and Victim Name of all crimes of "murder" and "sexual assault" that occurred in locations in "Lisbon".*

The following SQL queries are examples of queries that could correctly answer **[Task1]**.

Table 6 - Alternative Strategies to Solve **[Task1]**

Task	**Solution**
Task1_Alternative1	*Select crimeID, victimName* *From crimes* *Where crimeLocation IN (Select locationID From locations* *Where city = 'Lisbon') and crimeType IN ('murder', 'sexual assault')*
Task1_Alternative2	*Select crimeID, victimName* *From crimes Inner Join locations* *On crimeLocation = locationID* *Where city = ' Lisbon' and* *(crimeType = 'murder' or crimeType = 'sexual assault')*

To solve **[Task1],** there are at least two minimal subsets of skills the student can master, and they can be derived from the two solutions displayed in Table 6.
Task1_Alternative1 has two skills that Task1_Alternative2 does not have, they are: *"Find column values belonging to an explicit set (IN)"* and *"Find column values belonging to an sql select generated set (IN)"*.
Task1_Alternative2 has four skills Task1_Alternative1 does not have: *"Select data filtering rows using OR operator"*, *"Select data filtering rows using AND and OR operators (mix the two operators)"*, *"Understand the concept of Inner Join*" and "*Find matching rows in several tables (Inner Join)"*.

Task1_Alternative1 and Task1_Alternative2 are alternative ways of answering **[Task1]**. However, it is not correct to assume that alternative strategies are close by in the knowledge structure**.** Their location in the knowledge structure will depend on the precedencies of the skills associated to each alternative.

If this task is asked in an earlier phase of the assessment, the structure will be expecting the student to answer using Task1_Alternative1 as this is considered a more basic combination of skills. However, it is not impossible for the student to use Task1_Alternative2. If this happens, the algorithm will save the information about the skills the student showed mastery in, and his/her competency model will be updated accordingly.
In the end, the strategy chosen by the student will depend on the previously attained knowledge. Depending on that knowledge, the learner might answer **[Task1]** using one strategy but not the other. The other strategy might be in a further state of the structure.

Another example of skills that at first might be wrongly thought of as alternative strategies to master a task are *"Select data ordering the results in ascending order"* and *"Select data ordering the results in descending order"*. It is true that to master the skill *"Select the TOP rows of a result set",* usually one of these skills is used, but this does not imply that they are alternative strategies to master a task. Consider the following task.

[Task2]: *Write an SQL query to return the top 6 items with the lowest quantity in inventory.*

To solve the task above, the student will have to apply the skills *"Select the TOP rows of a result set"* and *"Select data ordering the results in ascending order"*. Applying *"Select data ordering the results in descending order"* is not a different strategy to solve **[Task2]**, as its outcome is not the same, and the result would be wrong.

3.3. IDENTIFY THE COMPETENCIES IN THE DOMAIN

Once the domain was defined and stage 1. in Figure 6 was completed, the identification of the domain's competencies/skills (stage 2. In Figure 6) was of importance. The question to ask at this point was "What must the students know to master the domain?".

To identify the skills in the domain, three books were used as reference. Korth et al. (2020) was the primary resource. This book has more extensive coverage of the SQL programming language when compared to the domain herein considered. The focus of the thesis is on chapters 3 and 4.1., that introduce the topics and sub-topics the students must know to master the domain under study. Two additional books were used to complement the previous, Batra (2018) and Amstrong (2016), serving as a double check after the first analysis. Also, the teaching experience of the experts was of value in this task.

For the skills identification process, the author and the expert independently gathered the set of skills they considered to cover the domain under study entirely. Through a discussion on both sets, an agreement was reached, and a merged set of competencies was defined. One of the challenges in identifying the competencies was the level of atomicity to use and the caution not to have different levels of atomicity within the set of competencies.

Having the skills identified, it was time to go back to Table 3 - The Six Categories of Cognitive Processes - in the literature review. At this point, an analysis was made having side by side the identified skills and the categories described in this table. The objective was to understand which skills would benefit from being split. In fact, all skills could be split into more skills, but as the categories in Table 3 are hierarchical, meaning that mastering an *Apply* skill entails *Understanding* it, and some skills are more straightforward the *Understand* option will just be considered for a minor set of skills. For example, the skill *"Select the TOP rows of a result set"* will not be split as it is considered not to implicate a very complex understanding, the most important part is to know how to apply it. On the other hand, the skill *"Find matching rows in several tables (Inner Join)"*, which is associated to the keyword *Inner Join* and is related to the verb *Apply*, will have an extra skill *"Understand the concept of Inner Join"* which will certainly be (one of) its antecedent(s).

The set of skills at this point was not yet the final set, as after filling out the task-competency matrix some skills were dropped. An effort was made to have all the skills represented in at least one task, but it was found that specific skills are not so common to be found in questions with the level of complexity of the domain under study, which does not cover all SQL programming language.

Consequently, the following skills were dropped: *"Find matching rows in several tables and keep the rows in the LEFT or RIGHT table that don't have a match (Full Outer Join)"*, *"Compare values from a column with values resulting from a subquery. Return true if the operation is true for any of the values*

in the subquery result (ANY)". Nevertheless, each of these skills have an associated *Understand* skill which remained in the set of competencies as their understanding is of value for mastering the domain.

Despite the effort not to have skills of different atomicity, in the stage of assigning the skills to the tasks it was necessary to come back to this step and discuss the split of three skills:

> **[Skill1]**: *"Select data filtering rows using AND, OR and NOT operators (does not include mixing AND and OR)"*
>
> **[Skill2]:** *"Selecting max/min/avg/sum/count values from a column table"*
>
> **[Skill3]:** *"Find matching rows in several tables including only the rows in the LEFT or RIGHT table that have a match (Left/Right Join)"*

While there were two skills regarding the *Order By* clause, one for ordering in an ascending manner and the other for ordering in a descendent manner, the three competencies enumerated above were mixing different concepts.

[Skill1] and **[Skill2]** were split into three and five skills, respectively, while **[Skill3]** remained the same. The reason for this difference is that while in **[Skill1]** and **[Skill2]**, each logical operator and aggregation function deliver different results (i.e. to get the maximum value of a column, the learner cannot use the *min()* function), regarding Right and Left join it is not quite the same. Even though the syntax of the Left and Right joins is different, the exact result of a Left join can be reached through a Right join.

As illustrated above, the process of identifying the competencies in the domain was iterative and was not closed until the filling of the task-competency matrix.

The full table of competencies, reached after the iterative process described, encloses 77 (seventy-seven) competencies, and can be found in Appendix A.

3.4. IDENTIFY THE TYPICAL TASKS IN THE DOMAIN

The identification and creation of the tasks is one of the most important stages in the entire process because the tasks and their competencies' assignment will dictate the structure of the competence and performance spaces.

Before getting to the tasks' creation, there is a series of interrogations that need to be answered. First, to create a task, it is important to know what types of questions may be asked (i.e., open-ended, true/false, multiple choice).

When thinking about the different kinds of questions one can use to evaluate students' knowledge on a domain, we must go further than the types of questions themselves and their advantages and disadvantages.

It is not productive to decide on the types of questions to ask without previously thinking about the platform that will be used, as it might have limitations that must be considered in the final decision.

Creating a complete solution to store the tasks and skills, evaluate students' responses and perform the knowledge assessment (i.e. an Intelligent Tutoring System) is completely out of the scope of the thesis. The main purpose of the needed supporting platform is the easy access from students and a

feature allowing the questioning of students and the collection of their responses. These two features allied with a procedure to evaluate students' responses, which will be discussed later, would be enough to perform the extra stage "Validate the structures with student data", referred to in the context of Figure 6 - Methodology Diagram.

Considering the main goal mentioned above, the Moodle platform seems to be a good choice. It is a Learning Management System (LMS) that presents, among many other features, the possibility to create quizzes and collect students' responses. Additionally, Moodle is the most popular e-learning platform in many regions (Matijašević-Obradović et al., 2017), and it is already in use in many portuguese higher education institutions (Rodrigues et al., 2018). The students and teachers are familiar with it, which makes it easier to collect the necessary data. Some examples of portuguese universities using Moodle are *Universidade Nova de Lisboa*, *Universidade de Lisboa*, *Universidade de Coimbra* and *Universidade Católica Portuguesa*. Globally speaking, Moodle is used by over 60% of all higher education[12].

3.4.1. Types of Questions

Within the chosen platform several types of questions are available: Multiple Choice, True/False, Matching, Numerical, Essay, Drag and Drop, Select Missing Words and others. There are even some additional question types that can be made available through a plugin. However, there will be no need to follow that path as it would make it harder for schools to implement the quiz and the default question types have all options that are needed.

In the identified competencies two types of learning objectives were defined. The verbs translating these objectives are *Understand* and *Apply.* While *Apply* is about using a procedure in a given situation, *Understand* is associated to construct meaning from something. Accordingly, the same type of question will not be suitable for testing both categories. For the sake of simplicity, *Apply* and *Understand* competencies will be addressed as practical and theoretical knowledge, respectively.

When debating on the type of question to use it is imperative to think on how the answer will be evaluated because the type of question might be just what is needed but then the answer's evaluation may present worrying limitations. It is important to note that this evaluation should be automatic.

Open-Ended questions are one of the question types that give less tips to the students on the correct answer, because they have literally a blank page in front of them. To know if a student has mastered a particular competency, it is good to know if he/she can answer the task without any tips but at the same time it gets very difficult to evaluate the answer as it is almost impossible to predict what the student will write, and most specifically, how he/she will write it.

For this reason, for more theoretical questions, open-ended questions will not be an option and alternatives must be considered. However, for the practical questions, which in our case involve writing SQL code, the open-ended questions might be a possibility to consider, as will be seen in due course. Before diving into the practical questions, lets discuss the type of questions to use for testing the theoretical competencies.

[12] Source: https://moodle.com/solutions/higher-education/

It is difficult to find questions as easy to evaluate as True/False and Multiple Choice, but these questions have a major disadvantage, that is the high probability of lucky guesses. As mentioned in 2.7 lucky guesses can be discouraged by subtracting points from the final grade when an incorrect response is given, but the objective of this thesis is not to grade the students, it is to understand what they know without putting on their shoulders the weight of a grade that will impact their final performance in the subject. Hence, True/False questions are discarded.

In Multiple Choice, the lucky guesses are also a problem, but while in True/False there only two alternatives, in multiple choice there are usually more. Regarding the number of options in multiple choice questions, having too many, despite reducing the chance of a lucky guess is not viable for the reasons mentioned in 2.7. The literature reviewed indicated that three was the optimal number but the difference in reliability between three and four options was found not to be high according to multiple studies.

Considering all the advantages and disadvantages discussed, multiple choice questions will be used for more theoretical questions.

Knowing that some multiple-choice questions will be reused from an existent question bank, which questions have four options, that will be the number of options in the multiple-choice tasks in this work.

Something discussed in the section 2.7 was the existence of an "I don't know" option. This will be put in practice as a way of reducing lucky guesses, but also to allow students to tell the knowledge assessment tool they do not know how to answer and consequently, have not mastered the respective competencies. This is important not only in multiple-choice questions but in all the tasks asked. When there is a grade that will be considered in the final performance in a class it is normal that the students do not want to say they do not know which is the right answer and prefer to take the risk of a lucky guess, but knowing this is not the case, this risk drops significantly, and a more honest response can be expected from the students. Here, it becomes clear the importance of explaining the students the context of the quiz.

Regarding practical questions, the discussion will be focused on the hypothesis of open-ended questions. If the student is asked to perform a query, he/she is expected to know how to write it from start to end and there is no better way of knowing if the student can do that than offering him/her a blank page to do it. The question that arises is the same as before "Is it possible to automatically evaluate the answer?".

As mentioned before, in the domain under study it is impossible to predict *a-priori* all the code variances capable of solving a task, there are different valid sets of words that make up a query capable of answering the same task. Even if an effort is made to account for the maximum alternatives possible, what if a new one, completely out of the box appears? That is a risk that should not be taken. However, discarding the open-ended questions for practical questions is not an option.

3.4.2. How To Compare Two SQL Queries

It is known that answers to practical questions will be done entirely in SQL. In the created solution, an SQL relational model is used to store correct answers to each task and student answers to the same tasks. The evaluation of the student answer is to be made by comparing the actual SQL code given by

the student with the correct answer stored in the database. The question is: How can two SQL queries be compared? There is not a direct answer to this question, but two alternatives are possible:

1. Semantic Comparison
2. Keywords Comparison
3. Direct Result Comparison

According to Oxford Advanced Learner's Dictionary, semantics is "the study of the meaning of words and phrases". So, the semantic analysis is more than the words themselves. It is about their meaning and the result they produce. The concern is not that the student does not answer exactly as the considered "right answer" as far as his/her answer delivers the same result no matter the dataset used. This answer can be reached through a semantic analysis of the two queries. Much work has been done (Chu et al., 2017; Murphy et al., 2018) in order to understand how to conclude about the semantic equivalence of SQL queries, but it was not until recently that some automated algorithms were created to perform this work. One example is named Cosette, and according to the creators "Given two queries Q_1 and Q_2, the query equivalence problem asks whether Q_1 and Q_2 are semantically equivalent, i.e., they always return the same results when executed on any input database instance." (Chu et al., 2017). This algorithm still has many limitations as the subset of syntax covered is rather small and implies not only pure SQL code evaluation but also some additional scripting – defining schemas, tables and the two queries each time a comparison is to be made. Some limitations are related to the syntax used to attribute alias to columns and tables, the way of referencing a table attribute or even features that are not covered yet, such as having expressions in the Group By clause[13]. It would make no sense to limit the way students write their code by what is accepted by the tool used, making Cosette inappropriate for the task at hand.

Regarding the keywords comparison option, one could think about parsing the student's answer to verify if the expected words exist in their query. Microsoft ScriptDom[14] parser would be a tool to explore for the parsing if this path is chosen. Although, as mentioned before, there is no single answer to a question, and it is not possible to know in advance all the sets of keywords we would have to look for, making this an unviable path.

As a result of the above comments, the third option – direct result comparison – will be the one to implement. After some research on this matter, no algorithm or software was found to address this issue. The only way to move forward was creating something from scratch that would allow the comparison of results obtained by running the two SQL queries under analysis. This approach has some limitations, but compared with the other options available, these are considered minor, and there are some ways of minimizing their impact.

The most evident limitation is that if the only focus are the queries' results, it is possible that by mere luck, the results are the same, even if the student's answer is not totally correct. This is true, but the chances of happening can be minimized. The most likely situation for this error to occur is when a specific question delivers an empty result set. A wrong answer by the student can also easily end up in an empty result set, and consequently, the two results would be equal. This situation will not occur if

[13] http://cosette.cs.washington.edu/guide

[14] https://devblogs.microsoft.com/azure-sql/programmatically-parsing-transact-sql-t-sql-with-the-scriptdom-parser/

it is guaranteed that each question has at least one record as a solution. As an example, if the question is: “How many employees did the company have in 2020?” then there would be at least one employee in this condition. It is not assured that the student’s answer delivers the same result as ours no matter the dataset used, but the evaluation will check if it delivers the same result using the same dataset.

The other limitation is that a student’s answer is only considered correct if the results are the same. In other words, if there is only a minor mistake in the query, leading to a different result, then it is considered wrong. Despite its limitations, the direct result comparison has the least impactful limitations compared to the other options – the algorithm for semantic comparison is in a very early stage of development and the keyword comparison is not adequate as it is impossible to know a-priori all the alternative queries to solve a task.

3.4.2.1. Direct Result Comparison Implementation

Since the queries to be evaluated are written in SQL, it was decided that the code created for this purpose would be done in SQL Server, using the Microsoft SQL Server Management Studio. A brief description of the created solution can be found below, and all the code is available in https://github.com/ajose98/Thesis_KST, under the folder “Comparing two SQL queries”. All the code is commented to ease its understanding.

First, a database containing multiple schemas was created. All schemas except one represent question statements (exercises) and store the data to be queried in each question. For better understanding of the content in these schemas, the figure below illustrates one of these schemas in the form of a physical diagram.

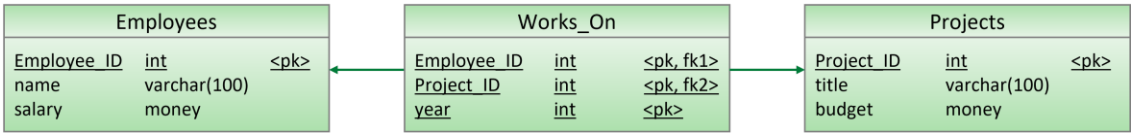


Figure 7 - Example of Question Schema

If the task “*What is the highest budget of a project?*" was to be asked, the solution given by the student would be tested against a correct solution, using the data in table *Projects*.

The last schema (“Solutions”) has a different purpose and has the following physical diagram.

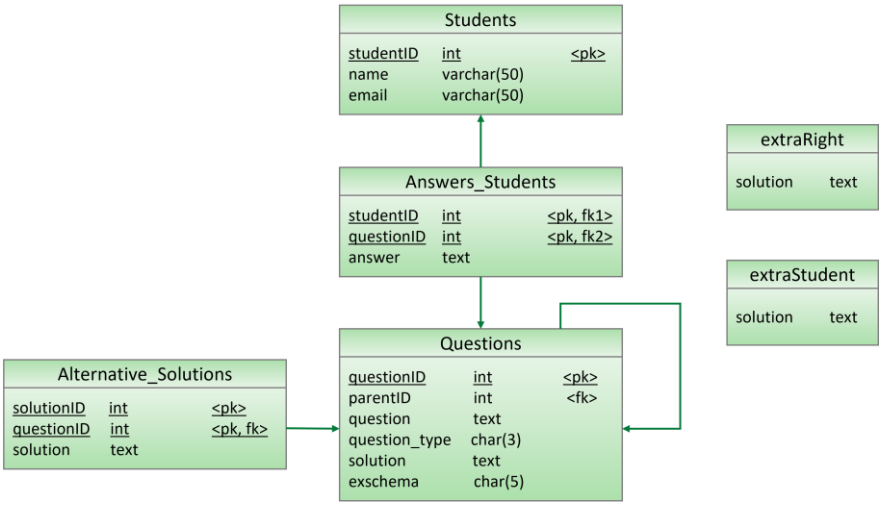


Figure 8 - Solutions Schema

The table *Questions* stores information on the questions, the *students* table has the information related to the students and the *Answers_Students* stores the answers of each student to each task. The *Alternative_Solutions* has information on the alternative ways of solving a question, and each question can have multiple alternative solutions. Tables *extraStudent* and *extraRight* are auxiliary tables that are used when evaluating an answer. For a better understanding of the attributes, a table with the most important attributes and their description is presented below.

Table 7 – Description of the Attributes in the Solutions Schema

Table	Attribute Name	Description
Answers_Students	*studentID*	Student identification number.
	questionID	Question identification number.
	answer	Student's answer (SQL query).
Questions	*questionID*	Question identification number.
	parentID	Each schema has a question statement to be presented in all questions of that schema, the parentID identifies the questionID which holds that common statement. It is null when the question is a parent of other questions.
	question	Question statement.
	question_type	Possible values: "I" (Insert in Existing table), "D" (Delete), "U" (Update), "INP" (Insert in New Permanent table), "INT" (Insert in New Temporary table).
	solution	Correct solution (SQL query). Is null when the question is the first of the schema, meaning it is the common statement.
	exschema	Denomination of the schema associated to the question.

To perform the evaluation, the correct solution (*solution*) and the student's answer (*answer*) are executed, and the results from both queries are separately concatenated from as many columns as they have into a single column, and then the comparison is made between the two text columns. If the columns have the exact same rows (regardless of the order), then the student's answer is determined as correct. The additional tables *extraRight* and *extraStudent* are used to store the concatenated results and they are cleaned after each student answer is evaluated.

Depending on the question type, the approach differs a little, but the common and most important part of the process is the one described above. In the end, three functions, four stored procedures, multiple triggers, and a file to compile it all and do the student's answer evaluation were created.

To get the results from the two queries, these queries must be used as subqueries, and this was a problem at first because the "Order By" clause cannot be used in subqueries, same as the "With" operator. For the "Order By" the solution was through the creation of two functions, which can be seen in the file *Functions&StoredProcedures.sql*. For the "With" operator and considering there are much more used alternatives for it, such as using a subquery in the From clause, no solution was created, and there would be a note at the beginning of the students' quiz saying they could not use this operator. This is seen as a minor limitation.

The phase after evaluating the student's answer would be getting for each correct answer the skills the student has used and showed mastery in. This could be achieved by having for each skill one (or more) regex expression(s) that identify it, i.e. if the student's answer includes the words "Select distinct", it means the student has used the skill "Select specific columns removing duplicates". The set of skills identified in the student's (correct) answer is then compared to the existing sets of skills assigned to that task in the structure.

3.4.3. Tasks Creation

Knowing the domain, the identified competencies, and the type of questions to ask, the tasks were created. Regarding the practical questions, all the tasks' statements and solutions were stored in the already mentioned "Solutions" schema. The theoretical questions are stored in a question bank in the Moodle platform.

The identification of the tasks was an iterative process, as displayed in Figure 6. The first set of tasks was identified before the assignment of skills to the tasks, but after completing the assignment, more tasks were added to the set so that all competencies had at least one associated task. After applying the precedence rule and analysing its results, it was not necessary to add more tasks, as all relations found were considered valid. However, if they were not valid, one counterexample task would have to be created for each of the invalid precedencies found. After stage 8. of the - Methodology Diagram it was also necessary to add tasks to the set. The reasoning for this will be given in chapter 3.8.

The tasks created after the first tasks' identification were appended to the existing task-competency matrix, and the respective skills were assigned.

Within the schemas, there are 135 (one hundred and thirty-five) different open response questions. Some of these tasks were adapted from questions in the question bank, and the others were inspired in the literature used to identify the skills in the domain.

Concerning theoretical questions, there are twenty-six tasks. Some were adapted from the same question bank; some were built using the literature in the domain. Whereas these twenty-six tasks cover only skills of the type *Understand*, the open response tasks cover *Understand* and *Apply* skills.

For all the tasks, the solutions were produced. There are nine open response questions with alternative solutions and one of them has four alternatives. All the question statements for open responses can be seen in Table 10 in Appendix B. Each question is identified by its ID, which corresponds to the number of the schema where they belong and the letter identifying the question in the schema. The multiple-choice questions can be seen in Table 11 in Appendix B. The solutions to these questions will not be made available to avoid compromising a future implementation of the structure in a knowledge assessment tool.

The 161 (one hundred and sixty-one) tasks were carefully created having in mind that both simpler questions and more complex tasks should exist in the structure, thus enabling the student to go from no knowledge in the domain to mastering the domain.

The process of creating and collecting the tasks was very time consuming. The final set of tasks is the most complete set it was possible to obtain, nevertheless it cannot be assumed as the complete set of tasks as the more questions there are, the more robust will the final structures be, just like in any data mining problem.

3.5. Assign Competencies to the Tasks

With the set of competencies and the set of tasks identified, the task-competency matrix can be constructed.

The columns represent the competencies, and the rows are the tasks. A reminder that multiple rows might be associated with the same task when there are alternative solutions.

The assignment of competencies to the tasks is made in a functional way, considering the solution of each task. This means that only the competencies that are directly used in the task are assigned to it.

The procedure of filling out the matrix was found to be time consuming and tiresome. As mentioned before, it was converted into an iterative process. After filling out the matrix for the first time, it was found that there were competencies with no tasks assigned, and some of the competencies were found not to be in the same level of atomicity. For the competencies with no tasks, there were two options: dropping those skills or creating tasks to address them. All but two competencies were maintained, and more tasks were produced. This reasoning was already described in 3.3.

Microsoft Excel was used to build and fill out the matrix. However, this was found not to facilitate the expert's task as it was not possible to visualize all the data at once since the matrix has too many columns and rows.

The table below contains some insights from the final filled-out matrix. The final matrix was not achieved until stage 8. of Figure 6 - Methodology Diagram.

Table 8 - Matrix Numerical Insights

	Minimum	Maximum	Average	Standard Deviation
Competencies per Task	1	18	6.4	4.0
Tasks per Competency	1	115	14.3	23.9

All tasks and competencies have at least one assignment, and the tasks with the most competencies have eighteen. This is the case for more complex tasks since they require a bigger number of skills. Remember that the skills are cumulative (i.e., if the *Apply* skill related to the *Inner Join* concept is required, so is its *Understand* skill). The competency with the most assignments is *"Select specific columns from a set of results"*, which is expected because this is the basis of any SQL query.

The standard deviation values are insightful since they indicate a bigger difference in the number of assignments for each competency than for each task. The standard deviation being higher than the average in the tasks per competency confirms that the standard deviation is extremely high for the scale in question. However, this was expected since there are basic competencies like *"Select specific columns from set of results"*, that are required more frequently to solve tasks and there are others not so commonly used or more complex, such as *"Insert data into an existing table using as input the result of a select statement"* and *"Match subquery table(s) with table(s) in the main query"*, which are only required in a more limited number of tasks.

The final matrix can be seen in Appendix C.

Besides its main role in the inference of the surmise relations between the competencies, this kind of matrix is also useful when preparing a course and creating questions for assessment tests. By identifying all the skills to test and the questions to ask, the teacher can easily verify if the created questions address all the intended competencies by doing the assignment of skills to the tasks. This procedure offers the professors a way of validating if the test covers all the expected content.

3.6. Inference Precedencies and Analyze the Results

Once the matrix was completed, we applied the **[Precedence Rule]** to the matrix to discover the surmise relations between the competencies.

To implement the precedence rule, a script was built in Python[15]. The Python script receives the Excel file with the matrix and outputs an Excel file with a table containing all the inferred surmise relations.

The outputted Excel file is the one used to perform stage 6. in Figure 6. Each row of the table represents a surmise relation of the kind "*x* is an antecedent of *y*", and each relation was validated by the experts.

The question the experts were told to ask themselves when evaluating the results was "Is it true that *x* should be learnt before *y*?". If the answer is negative, that relation is classified as spurious, and a

[15] All the Python code mentioned is available in https://github.com/ajose98/Thesis_KST, under the folder "Matrix Related Code"

counterexample task needs to be created to eliminate this precedency from the set of results. If the answer is positive, the relation is validated.

This validation process allies the functional precedencies derived by the algorithm, which are based on the tasks and their required competencies, with a more pedagogical approach given by the experts. If the set of tasks provides good domain coverage, all relations are expected to be valid.

After the individual validation, all the surmise relations were considered valid, and no extra tasks were created in this phase.

3.7. Derive the Base of the Competence Space

The number of feasible competence states obtained from the task-competency matrix are 147 (one hundred and forty-seven). Each of these subsets of competencies can solve at least one task in the set of tasks. However, this is not the complete competence space. The complete competence space could have a maximum of 2^{77} feasible states (Doignon & Falmagne, 1999), but the existence of precedencies lowers the number of possible states. Nevertheless, as the structure obtained through the method used is a space, there is no need to store the entire competence space, which would require a lot of storage capacity. If the base of the competence space is known, then the entire space can be computed from it at any time by closing the states in the base under union.

The base of the competence space is included in the 147 states obtained from the matrix, and it is possible to know which of the 147 states form the base: a state belongs to the base if it cannot be obtained through the union of two or more states in the base. First, duplicated subsets were removed and then, having the rule mentioned above, some code was written in Python to discover the base of the competence space.

The base was found to have 125 (one hundred and twenty-five) states, excluding the empty state, which by convention never belongs to a base (Doignon & Falmagne, 2016).

3.8. Derive the Performance Structure Associated to the Base

It is important to compute the performance states associated with the states in the base of the competence space to confirm if more tasks should be added to the set of tasks. According to the discussion in section 3.1 - Methods Comparison and Discussion, regarding the inclusion of alternative answers as independent tasks, it was discussed that if a state contains a single task and that task represents an alternative answer, then at least one more task must be added to that performance state to avoid that state becoming unreachable.

After computing the table in Appendix D and analysing it in the light of the last paragraph, it was found that the performance states in the rows with IDs 51, 61, 63, 72, 76, 80, 90, 115 and 123 need to have one more task each. Consequently, nine tasks were added to the set of tasks and to the task-competency matrix. These tasks are included in Table 10, in Appendix B, regarding the identified tasks in the domain.

In Appendix D the tasks' IDs are identical to the ones used in Appendix B, except for tasks that have alternative solutions, which were appended with '_x', where x corresponds to the number identifying

the alternative. Additionally, unlike the IDs for the practical questions, the IDs for the theoretical ones are completely numerical.

3.9. VISUALIZE THE BASE OF THE COMPETENCE SPACE

Even though the complete competence space was not computed, the base states, which reflect the entire space, can be displayed in a precedence graph to visualise and analyse the obtained competence space.

From the options discussed in section 2.8 - How to Visualize and Store a Knowledge Space, and considering the main purpose at this point – visualizing and validating the base of the Competence Space - the Gephi software was chosen to store and visualize the base of the competence space. If we were to use the Competence Space in a knowledge assessment tool, a graph database would be preferred because instead of Gephi, which is mainely a visualization tool, graph databases are meant to store data.

To build the precedence graph, two types of information were needed: the nodes of the graph – the base states – and the edges – the relations between the base states. The nodes were already discovered, but for the edges, some additional scripting in Python was required. To find the edges, the states which were subsets of others were determined and if a chain of precedencies was found, only the consecutive precedencies were maintained (e.g. state {1} is a subset of the states {1, 11} and {1, 11, 13}, but {1, 11} is also a subset of {1, 11, 13}, consequently, only the edges ({1}, {1, 11}) and ({1, 11}, {1, 11, 13}) were plotted in the precedence graph).

The information on the nodes and edges was stored in an Excel file which was used to build the visualization. The precedence graph was built using the open-source Gephi software and by applying the “Force Atlas” layout. As the graph has 125 nodes, it was exported as a pdf file to allow better visualisation and interaction with the graph, enabling its expansion. It can be seen in Appendix E.

In the precedence graph, when an arrow is pointing from node *x* to node *y,* it means that node *x* is a subset of *y*, or in KST terms, to know *y* the student must have mastered *x*. From a teaching point of view, this relation could be seen as “before presenting the student with the tasks associated to node *y* (and teaching those competencies), he/she should first learn the competencies associated with node *x* and solve its task(s)”.

4. RESULTS AND DISCUSSION

An overview of the visualization in Appendix E can be seen in Figure 9. This figure helps drawing conclusions about the obtained competence space, which is reflected in its base.

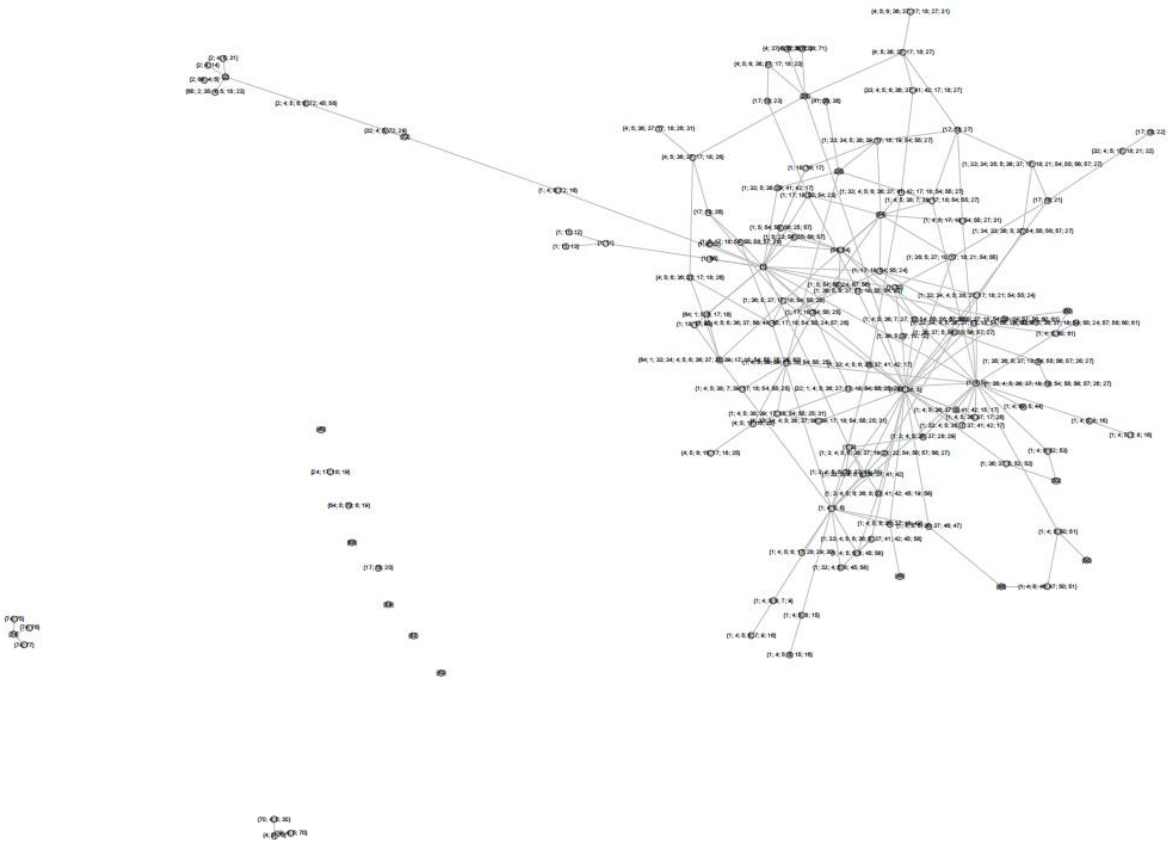


Figure 9 - Visualization of the Base of the Competence Space (overview)

The nodes are represented by small circles, and the label above each node identifies the competencies in each competence state/node.

The graph shows eight independent states – located in the center left of the figure –, meaning they are not related to any other node. A state of the base is independent if the set of competencies composing it is not a subset of any other state and if there is no state which is a subset of the independent state itself. However, when the competence space is computed, they will not be unrelated anymore. They will have no antecedent nodes, but they will have precedents resulting from the union of these states with all the others.

Five of the independent states are composed of only one competency. Two of these cases are “Insert data into a table using a list of values for all the attributes” and “Insert data into a table using a list of values for some attributes”. The independence of these two skills indicates that these skills do not interact with others in the tasks, which is a reasonable statement given the typical tasks in the domain. Consequently, they do not have to be taught or learnt in a particular order. Regarding the other independent states with only one competency, they are all *Understand* skills that do not have a corresponding *Apply* skill (e.g., “Understand the concept of Cross Join”). In this context, it is not so

reasonable to conclude that these skills are usually used independently, as some tasks may require more than one of these skills in order to be solved. In conclusion, some task(s) requiring these *Understand* skills and some other skill(s) should be added to the set of tasks to enrich it and make it more complete, resulting in a more robust and representative structure of the domain.

For the three independent states consisting of more than one competency, the most important conclusion to take is also related to the fact that there are other states with more than one competency (sometimes even more than four competencies, e.g., {4, 5, 17, 18, 25}) that have no precedent states. The existence of base states in this condition will result in a competence space that does not comply with the property of Learning Smoothness – implying the items can be learnt one by one. (see condition (3) in section 2.3): after closing the base states under union, the states with no precedent nodes will keep having no precedents. This property assures that the difference between the number of competencies in consecutive states in the network is always one. This is essential for the competence space to become a learning space (see section 2.3). In a competence space with Learning Smoothness the student will only have to master one competency at a time, what provides a more precise guidance to both students and teachers.

To reach a competence space compliant with the Learning Smoothness property, more less complex tasks must be added to the set of tasks. Using the example of state {4, 5, 17, 18, 25}, this state would have to have at least four precedent states to assure that the skills could be mastered one by one, guaranteeing Learning Smoothness. For those four states to be part of the structure, at least four tasks (one for each state) have to be added to the set of tasks.

Looking closely at the graph in Figure 9, one can note that there are certain states that have many arrows parting from it, meaning that these states represent the subsets of competencies most found in the tasks in the domain. The two states that are the source of more edges are states {1} and {1, 37, 36, 5}, which is in line with the expectation, as competency 1 is the most basic skill ("Select specific columns from set of results") and the usage of skill 1 with the Inner Join and filtering rows using comparison operators like $=, <, >=$ is also a very common combination to be found in many tasks in the domain.

One last note on the visualization is the existence of two clusters – located in the left side and bottom of the graph - completely detached from the central part of the graph. One of these clusters is composed of states which competencies regard a particular subject of the domain, which is the concept of keys (primary key, foreign key, composite primary key and alternate key). These is an essential set of skills in the domain, but they are more conceptual than the Apply skills, and the practical tasks do not directly require the need to master these skills. Therefore, as they only interact with each other, they do not appear in other states and have a cluster of their own. This indicates that this set of skills is independent of the other skills. We can argue that this sub-structure should be learnt before the main structure as these competencies, even though they are not a functional precedence of the practical tasks, they are formal concepts the students should know before solving the practical questions.

The other detached cluster contains all the non-independent states containing competency 70 ("Update values from a table using the information on it"). These states would not be distant from the rest if the Learning Smoothness property was present, as there are some subsets of these states that are present in other states.

One other cluster that is not completely detached from the bigger cluster but is quite far from it is located in the upper left of Figure 9. This cluster contains all the states comprising competency 2 ("Select all columns from the set of results"). The distance exists because the number of tasks requiring this skill is not huge, as the most common in an SQL statement is to select specific columns from the result set.

A conclusion to draw from the analysis and discussion of the results obtained in the base of the competence space is the need to refine the competence space before implementing it in a knowledge assessment tool. Despite the iterative nature of the implemented method, leading to a step-by-step validation of the results, there are still some gaps in the identified tasks. Identifying the typical tasks in the domain is a very tiresome job, and like in any data mining project, most tasks are identified, the better the result will be. The creation of an algorithm to check the states that should exist for the Competence space to be smooth is a way to go to help the experts identifying the missing tasks. More about this will be said in the chapter Future Work.

The base of the competence space was the final outcome of this work, but some intermediary steps made to get to this outcome are themselves insightful tools for the teaching process. The task-competency matrix can help the teachers ensure that all the competencies they want to test are covered by the tasks asked. Also, the SQL solution created to compare two SQL queries (see more in section 3.4.2.1) can be integrated with a questioning platform and can automatically evaluate the students' answers. Additionally, in case of an incorrect answer, the student could be given feedback by showing him/her the difference between the set of results of the student's answer and the expected set of results.

5. CONCLUSIONS

The proposed work of this thesis was to create a structure that could map the knowledge inside a restricted part of the SQL domain. This was achieved through the study and application of the Knowledge Space Theory and its related concepts. This kind of structure enables professors to improve their teaching approaches and to better follow the students' progress by performing more insightful knowledge assessments.

The literature review led to the comparison of four methods that start from individual items and result in a knowledge space. Depending on each method, the meaning of the items in the structure could differ. However, the chosen method was an approach based on an extension of KST, called Competence-Based KST, which results in two structures – the competence space and the performance structure - instead of one like in the other approaches. In this method, both the competencies and the tasks in the domain are considered to create the structures, and both are derived from the set of identified tasks in the domain. The competence space is a knowledge space.

The final competence space is delivered in the form of its base, a subset of all states in the domain that can generate all the others. This subset of states that can generate the entire competence space is the main outcome of this work. With the entire competence space and the matrix used to assign the tasks in the domain to the respective skills (see Appendix D), the entire performance structure can be generated, and the two structures – the competence space and the performance structure - could be used in a knowledge assessment tool. However, as discussed in section "Future Work and Limitations", after analysing the base of the competence space using the visualization in Appendix E, it was found that the achieved structures would benefit from additional work in order to transform the competence space in a learning space (see this in more detail in section "Knowledge Spaces vs Learning Spaces").

Besides the main outcome – the base of the competence space -, some intermediary steps led to useful outcomes in the domain of SQL. Two important questions were answered "What are the types of problems/competencies that exist in this particular domain?" and "What are the typical tasks/problems in the domain?". The answers to these questions can be seen in Appendix A and Appendix B, respectively. The assignment of the competencies to the identified tasks is also of value to any professor of SQL as it shows what is the domain content being cover in each task. This assignment was done through a task-competency matrix and it is available in Appendix C.

The states in the base of the competence space can be seen in Appendix D, along with the associated states in the performance structure, and its visual representation can be seen in more detail in Appendix E.

The methodology implemented had its basis on the method reviewed, but some add-ons were made to improve the final structures. It became an iterative method and allowed the existence of alternative ways of solving a question. The innovation also comes from the lack of literature concerning knowledge spaces and the domain under study.

6. FUTURE WORK AND LIMITATIONS

The refinement of the competence space is a necessary future work to have a space that represents the domain in the best manner possible. To refine the competence space obtainable from the base, it will be necessary to first compute the entire competence space by closing under union all the base states. This procedure demands a very high storage capacity as the space will have many states.

After having the complete competence space, the properties of Learning Smoothness – implying the items can be learnt one by one. (see condition (3) in section 2.3) - and Learning Consistency - implying that knowing more does not prevent learning something new. (see condition (4) in section 2.3) - must be confirmed. In the domain under study and considering the identified competencies, it is viable to say that all competencies can be mastered one by one, so Learning Smoothness is a meaningful property and allows for a more precise knowledge assessment. Regarding Learning Consistency and the domain, it is true that knowing more should not prevent learning something new.

To validate Learning Smoothness, we should iterate through all the states in the space. The process could start in the empty state by verifying if all states which are its direct precedents have exactly one competency. Then, from all the states that have one competency, verify if their direct precedents have exactly two items and so on. If the difference between two consecutive states is more than one competency, the states in between are missing. These missing states have to be computed, and the experts must create tasks that feature the competencies in these missing states so that they are present in the structure when the algorithm is rerun. This process will result in a more complete set of tasks.

To validate Learning Consistency, a more complex algorithm must be created. For all competence states that satisfy the condition $K \subset L$ and q is a competency such that $K + \{q\}$ is a feasible state, then one must confirm that $L + \{q\}$ is also a state of the structure. If $L + \{q\}$ is not a feasible state, its inclusion should be considered, and a task that features the competencies in L and competency q must be created, resulting in a more complete set of tasks.

After adding the tasks needed to assure the two properties mentioned above, the base needs to be updated. The two algorithms described above would need high computational power.

After this first refinement based on the two learning properties, it would be insightful to validate the structure using student data. This could be done by presenting to a student a set of tasks spread in the structure and then applying the leave one out method, as suggested by Ley et al. (2010), to find if the student's answers to all tasks but one, along with the structure can predict the outcome (correct or incorrect) of the student's answer to the left out question. Other validation methods using students' data should be discussed.

The process of identifying the competencies, tasks and filling the task-competency matrix was found to be the more tiresome part of this work and it can discourage someone who wants to implement the methodology described in other domains. Consequently, the creation of a user-friendly program that eases these tasks and integrates them in a manner that new tasks can be added on-demand, and existing or new competencies can be assigned to them, resulting in an update of the existing structures would be of great value.

The most outstanding limitations encountered during this work were the subjectivity of the data collection process (the identification of competencies and tasks) and the small number of experts (two) involved in the process.

7. BIBLIOGRAPHY

OECD (2013), Synergies for Better Learning: An International Perspective on Evaluation and Assessment, OECD Reviews of Evaluation and Assessment in Education. *OECD Publishing, Paris*, https://doi.org/10.1787/9789264190658-en.

Reay, D., & Wiliam, D. (1999). "I'll be a nothing": Structure, agency and the construction of identity through assessment [1]. *British Educational Research Journal*, *25*(3), 343–354. https://doi.org/10.1080/0141192990250305

Clarke, M., Shore, A., Rhoades, K., Abrams, L. M., Miao, J., & Li, J. (2003). Perceived Effects of State-Mandated Testing Programs on Teaching and Learning: Findings from Interviews With Educators in Low-, Medium-, and High-Stakes States. *Methodology*, *January*, 106.

Albert, D., Doble, C., & Eppstein, D. (2013). *Knowledge Spaces : Applications in Education*.

Albert, D., Mayer, B., & Heller, J. (2005). Competence-based Knowledge Structures for Personalised Learning. *International Jl. on E-Learning*, *5*(1), 75–88.

ALEKS Corporation. (2013). *ALEKS Instructor's Manual for Math Prep for Accounting*.

Alias, M., & Gray, D. E. (2005). The Learning Hierarchy Technique: An Instructional Analysis Tool in Engineering Education. *Australasian Journal of Engineering Education*, *04*(June 2014), 1963.

Amstrong, D. (2016). *SQL: with practice exercises, Learn SQL Fast, on free software*.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's Taxonomy of educational objectives*. Addison Wesley Longman, Inc.

Batra, R. (2018). *SQL Primer - An Accelerated Introduction to SQL Basics*.

Baumunk, K., & Dowling, C. E. (1997). Validity of Spaces of Assessing Knowledge about Fractions. *Journal of Mathematical Psychology*, *41*, 99–105.

Burton, R. F. (2002). Misinformation, partial knowledge and guessing in true/false tests. *Medical Education*, *36*(9), 805–811. https://doi.org/10.1046/J.1365-2923.2002.01299.X

Bush, M. E. (2006). Quality assurance of multiple-choice tests. *Quality Assurance in Education*, *14*(4), 398–404. https://doi.org/10.1108/09684880610703974

Chu, S., Wang, C., Weitz, K., & Cheung, A. (2017). Cosette : An Automated Prover for SQL. *8th Biennial Conference on Innovative Data Systems Research*.

Clarke, M., Shore, A., Rhoades, K., Abrams, L. M., Miao, J., & Li, J. (2003). Perceived Effects of State-Mandated Testing Programs on Teaching and Learning: Findings from Interviews With Educators in Low-, Medium-, and High-Stakes States. *Methodology*, *January*, 106.

de Chiusole, D., Stefanutti, L., & Spoto, A. (2017). A class of k-modes algorithms for extracting knowledge structures from data. *Behavior Research Methods*, *49*, 1212–1226. https://doi.org/10.3758/s13428-016-0780-7

Dehnad, A., Nasser, H., & Hosseini, A. F. (2014). A Comparison between Three-and Four-Option Multiple Choice Questions. *Procedia - Social and Behavioral Sciences*, *98*, 398–403. https://doi.org/10.1016/J.SBSPRO.2014.03.432

Desmarais, M. C., Meshkinfam, P., & Gagnon, M. (2006). Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction*, *16*(5), 403–434. https://doi.org/10.1007/s11257-006-9016-3

Doble, C., Thiéry, N., Uzun, H., Cosyn, E., & Falmagne, J. C. (2006). *Assessing Mathematical Knowledge in a Learning Space : Validity and / or Reliability** (Issue 949).

Doignon, J. P., & Falmagne, J. C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, *23*(2), 175–196. https://doi.org/10.1016/S0020-7373(85)80031-6

Doignon, J. P., & Falmagne, J. C. (1999). *Knowledge Spaces*. Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35329-1

Doignon, J. P., & Falmagne, J. C. (2011). Learning Spaces: Interdisciplinary Applied Mathematics. In *Springer, Berlin, Heidelberg*.

Doignon, J. P., & Falmagne, J. C. (2016). Knowledge spaces and learning spaces. *New Handbook of Mathematical Psychology*, *1*, 274–321. https://doi.org/10.1017/9781139245913.006

Doroudi, S., Holstein, K., Aleven, V., & Brunskill, E. (2016). Sequence matters, but how exactly? A method for evaluating activity sequences from data. *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016*, 70–77.

Dowling, C. E., Koch, U., & Quante, K. A. (1996). A new interface for querying experts on prerequisite relationships. *Proceedings - 6th Australian Conference on Computer-Human Interaction, OzCHI 1996*, 320–321. https://doi.org/10.1109/OZCHI.1996.560150

Falmagne, J. C., Cosyn, E., Doignon, J. P., & Thiéry, N. (2006). The assessment of knowledge, in theory and in practice. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 3874 LNAI* (Issue 949). https://doi.org/10.1007/11671404_4

Falmagne, J. C., Doignon, J. P., Koppen, M., Villano, M., & Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, *97*(2), 201–224. https://doi.org/10.1037/0033-295X.97.2.201

Foraita, R., Spallek, J., & Zeeb, H. (2014). Directed Acyclic Graphs. *Handbook of Epidemiology: Second Edition*, 1481–1517. https://doi.org/10.1007/978-0-387-09834-0_65

Guo, L., Wang, D., Gu, F., Li, Y., Wang, Y., & Zhou, R. (2021). Evolution and trends in intelligent tutoring systems research: a multidisciplinary and scientometric view. *Asia Pacific Education Review*. https://doi.org/10.1007/s12564-021-09697-7

Hamtini, T., Albasha, S., & Varoca, M. (2015). Towards Designing an Intelligent Educational Assessment Tool. *Journal of Software Engineering and Applications*, *8*, 35–42. https://doi.org/10.4236/jsea.2015.82005

Hartley, J. R., & Sleeman, D. H. (1973). Towards more intelligent teaching systems. *International Journal of Man-Machine Studies*, *5*(2), 215–236. https://doi.org/10.1016/S0020-7373(73)80033-1

Higher Education Math Placement. (2016).

Kambouri, M., Koppen, M., Villano, M., & Falmagne, J.-C. (1994). Knowledge assessment: tapping human expertise by the QUERY routine. *International Journal of Human-Computer Studies*, *40*(1),

119–151.

Kang, S. H. K., McDermott, K. B., & Roediger, H. L. (2007). Test format and corrective feedback modify the effect of testing on long-term retention. *European Journal of Cognitive Psychology*, *19*(4–5), 528–558. https://doi.org/10.1080/09541440601056620

Kickmeier-Rust, M. D., & Albert, D. (2015). Visualizing the Structure of Learning. *CEUR Workshop Proceedings (Vol. 1599)*.

Knaus, K., Murphy, K., Blecking, A., & Holme, T. (2011). A valid and reliable instrument for cognitive complexity rating assignment of chemistry exam items. *Journal of Chemical Education*, *88*(5), 554–560. https://doi.org/10.1021/ed900070y

Koppen, M. (1993). Extracting human expertise for constructing knowledge spaces: an algorithm. *Journal of Mathematical Psychology*, *37*, 1–20.

Koppen, M., & Doignon, J. P. (1990). How to build a knowledge space by querying an expert. *Journal of Mathematical Psychology*, *34*(3), 311–331. https://doi.org/10.1016/0022-2496(90)90035-8

Kornell, N., Hays, M. J., & Bjork, R. A. (2009). Unsuccessful Retrieval Attempts Enhance Subsequent Learning. *Journal of Experimental Psychology: Learning Memory and Cognition*, *35*(4), 989–998. https://doi.org/10.1037/a0015729

Korossy, K. (1999). Modeling Knowledge as Competence and Performance. In *Knowledge Spaces: Theories, Empirical Research, and Applications* (pp. 103–132).

Korth, H. F., Silberschatz, A., & Sudarshan, S. (2020). *Database System Concepts* (Seventh Ed). McGraw-Hill Education. https://doi.org/10.1145/253671.253760

Lazarska, M., & Siedlecka-Lamch, O. (2019). Comparative study of relational and graph databases. *INFORMATICS 2019 - IEEE 15th International Scientific Conference on Informatics, Proceedings*, *November 2019*, 363–370. https://doi.org/10.1109/Informatics47936.2019.9119303

Lee, F.-L., & Heyworth, R. (2000). Problem Complexity: A Measure of Problem Difficulty in Algebra by Using Computer. *Education Journal-Hong Kong-Chinese University of Hong Kong*, *28*(1), 85–108.

Ley, T., Kump, B., & Albert, D. (2010). A methodology for eliciting, modelling, and evaluating expert knowledge for an adaptive work-integrated learning system. *International Journal of Human Computer Studies*, *68*(4), 185–208. https://doi.org/10.1016/j.ijhcs.2009.12.001

Liu, Z. Y., Lomovtseva, N., & Korobeynikova, E. (2020). Online learning platforms: Reconstructing modern higher education. *International Journal of Emerging Technologies in Learning*, *15*(13), 4–21. https://doi.org/10.3991/ijet.v15i13.14645

Marte, B., Steiner, C. M., Heller, J., & Albert, D. (2008). Activity- and taxonomy-based knowledge representation framework. *International Journal of Knowledge and Learning*, *4*(2–3), 189–202. https://doi.org/10.1504/IJKL.2008.020654

Matijašević-Obradović, J., Dragojlović, J., & Babović, S. (2017). *The Importance of Distance Learning and the Use of Moodle Educational Platform in Education*. *January*, 236–241. https://doi.org/10.15308/sinteza-2017-236-241

Murphy, B., Chu, S., Roesch, J., Cheung, A., & Suciu, D. (2018). Axiomatic foundations and algorithms for deciding semantic equivalences of SQL queries. *Proceedings of the VLDB Endowment*, *11*(11), 1482–1495. https://doi.org/10.14778/3236187.3236200

Nasar, S. (2016). *Knowledge Space Framework: An API for representation, persistence and visualization of knowledge spaces* (Issue May).

Nwaogu, E. (2013). The effect ALEKS of on students' mathematics achievement in an online learning environment and the cognitive complexity of the initial and final assessments. In *Dissertation Abstracts International Section A: Humanities and Social Sciences* (Vol. 73, Issues 12-A(E)).

Raymond, M. R., Stevens, C., & Bucak, S. D. (2019). The optimal number of options for multiple-choice questions on high-stakes tests: application of a revised index for detecting nonfunctional distractors. *Advances in Health Sciences Education*, *24*(1), 141–150. https://doi.org/10.1007/s10459-018-9855-9

Reay, D., & Wiliam, D. (1999). "I'll be a nothing": Structure, agency and the construction of identity through assessment [1]. *British Educational Research Journal*, *25*(3), 343–354. https://doi.org/10.1080/0141192990250305

Robinson, P. (2001). Task complexity, task difficulty, and task production: Exploring interactions in a componential framework. *Applied Linguistics*, *22*(1), 27–57. https://doi.org/10.1093/applin/22.1.27

Rodrigues, S., Rocha, A., & Abreu, A. (2018). Needs and expectations faced with the Moodle platform and institutional support available: The case of ISCAP. *Iberian Conference on Information Systems and Technologies, CISTI*, *2018-June*, 1–6. https://doi.org/10.23919/CISTI.2018.8399409

Scalise, K., Bernbaum, D. J., Timms, M., Harrell, S. V., Burmester, K., Kennedy, C. A., & Wilson, M. (2007). Adaptive Technology for E-Learning: Principles and Case Studies of an Emerging Field. *Journal of the American Society for Information Science and Technology*, *58*, 2295–2309. https://doi.org/10.1002/asi

Segedinac, M., Savić, G., Konjović, Z., & Segedinac, M. (2010). Optimal counterexamples expectation based method for knowledge space construction. *SIISY 2010 - 8th IEEE International Symposium on Intelligent Systems and Informatics*, 273–278. https://doi.org/10.1109/SISY.2010.5647458

Segedinac, M. T., Horvat, S., Rodić, D. D., Rončević, T. N., & Savić, G. (2018). Using knowledge space theory to compare expected and real knowledge spaces in learning stoichiometry. *Chemistry Education Research and Practice*, *19*(3), 670–680. https://doi.org/10.1039/c8rp00052b

Shrier, I., & Platt, R. W. (2008). Reducing bias through directed acyclic graphs. *BMC Medical Research Methodology 2008 8:1*, *8*(1), 1–15. https://doi.org/10.1186/1471-2288-8-70

Shute, V. J., & Zapata-Rivera, D. (2010). Intelligent Systems. In *International Encyclopedia of Education*. Elsevier. https://doi.org/10.1016/B978-0-08-044894-7.00247-5

Stahl, C., & Hockemeyer, C. (2019). *Knowledge Space Theory*. 1–10.

Ünlü, A., & Sargin, A. (2010). DAKS: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, *37*(2), 1–31. https://doi.org/10.18637/jss.v037.i02

Vegada, B., Shukla, A., Khilnani, A., Charan, J., & Desai, C. (2016). Comparison between three option, four option and five option multiple choice question tests for quality parameters: A randomized study. *Indian Journal of Pharmacology*, *48*(5), 571–575. https://doi.org/10.4103/0253-7613.190757

Vicknair, C., Nan, X., Macias, M., Chen, Y., Zhao, Z., & Wilkins, D. (2010). A comparison of a graph database and a relational database: A data provenance perspective. *Proceedings of the Annual*

Southeast Conference, *August 2016*. https://doi.org/10.1145/1900008.1900067

Vyas, R., & Supe, A. (2008). Multiple choice questions: A literature review on the optimal number of options. *National Medical Journal of India*, *21*(3), 130–133.

8. APPENDIX A

Table 9 - Identified Competencies

CompetencyID	Competency	Learning Objective
1	Select specific columns from set of results	Apply
2	Select all columns from set of results	Apply
3	Select specific columns removing duplicates	Apply
4	Filter rows using conditions in Where clause	Apply
5	Select data filtering rows using simple comparison operators (=,<,>,=<,=>,!=, BETWEEN)	Apply
6	Select data using AND operator	Apply
7	Select data using OR operator	Apply
8	Select data using NOT operator	Apply
9	Select data using AND and OR operators (mix the two operators)	Apply
10	Select data ordering the results in ascending order	Apply
11	Select data ordering the results in descending order	Apply
12	Select the TOP rows of a result set	Apply
13	Select the TOP percentage of a result set	Apply
14	Find patterns in a column of a table (LIKE)	Apply
15	Find column values belonging to an explicit set (IN)	Apply
16	Find column values belonging to an sql select generated set (IN)	Apply
17	Manipulate column titles in result sets (AS)	Apply
18	Create a calculated column	Apply
19	Use arithmetic expressions	Apply
20	Concatenate multiple column values	Apply
21	Get maximum value(s) from a column table	Apply
22	Get minimum value(s) from a column table	Apply
23	Get mean value(s) from a column table	Apply

24	Get sum value(s) from a column table	Apply
25	Get count value(s) from a column table	Apply
26	Get count values from a column table removing duplicates from a specific column	Apply
27	Get count values from a table considering null values	Apply
28	Compare date values	Apply
29	Get the current date	Apply
30	Add time to a date	Apply
31	From a date value take only a part of it	Apply
32	Get the difference between two dates	Apply
33	Manipulate table names in queries (AS)	Apply
34	Use subqueries to return a set of results in the From clause	Apply
35	Use scalar subqueries for filtering rows in the Having and/or Where clauses	Apply
36	Understand the concept of Inner Join	Understand
37	Find matching rows in several tables	Apply
38	Understand the concept of Left/Right Join	Understand
39	Find matching rows in several tables including only the rows in the LEFT or RIGHT table that have a match	Apply
40	Understand the concept of Full Outer Join	Understand
41	Understand the concept of Self Join	Understand
42	Compare rows from the same table	Apply
43	Understand the concept of Cross Join	Understand
44	Use scalar subqueries in the select statement	Apply
45	Match subquery table(s) with table(s) in the main query	Apply
46	Understand the concept of Union	Understand

47	Unite sets taken from different tables having the same structure (UNION)	Apply
48	Understand the concept of Union All	Understand
49	Unite sets taken from different tables having the same structure (UNION ALL) keeping the duplicate values	Apply
50	Understand the concept of Intersect	Understand
51	Intersect sets taken from different tables having the same structure (INTERSECT)	Apply
52	Understand the concept of Except	Understand
53	Show elements of a set that don't exist in another set (EXCEPT)	Apply
54	Understand the concept of Group By	Understand
55	Group values taken from a table (or list of tables)	Apply
56	Understand the purpose of Having after the Group By clause	Understand
57	Group values taken from a table (or list of tables) and impose conditions on the GROUP BY result set (HAVING)	Apply
58	Select the existence of records in a result set from a subquery (EXISTS)	Apply
59	Understand the functioning of the ANY operator	Understand
60	Understand the functioning of the ALL operator	Understand
61	Compare values from a column with values resulting from a subquery. Return true if the operation is true for all values in the subquery result	Apply
62	Test for null/non null values	Apply
63	If a value is null return something else	Apply
64	Impose conditions on the result set (Case When)	Apply
65	Create a new persistent table from the result of a SELECT	Apply
66	Create a new temporary table from the result of a SELECT	Apply

67	Insert data into a table using a list of values for all the attributes	Apply
68	Insert data into a table using a list of values for some attributes	Apply
69	Insert data into an existing table using as input the result of a select statement	Apply
70	Update values from a table using the information on it	Apply
71	Update values from a table using information from other tables	Apply
72	Delete rows of a table based on some condition(s) using its own information	Apply
73	Delete rows of a table using information from other tables	Apply
74	Understand the concept of single Primary Key	Understand
75	Understand the concept of composite Primary Key	Understand
76	Understand the concept of Foreign key	Understand
77	Understand the concept of alternate key	Understand

9. APPENDIX B

Table 10 - Identified Tasks (Open response)

ID	Question Statement
1	ex001. Consider the following relational model: EMPLOYEES(EMPLOYEE_ID, name, salary) PROJECTS(PROJECT_ID, title, budget) WORKS_ON(EMPLOYEE_ID, PROJECT_ID, year) EMPLOYEES stores information about the employees of a company and PROJECTS stores information regarding projects developed by the company. The WORKS_ON table contains information about the projects in which employees collaborated. Name is the name of the employee and salary keeps information about the employee's annual salary title is the project title and budget keeps information about the project budget. An employee can work on several projects during the same year and on the same project in different years (if this project lasts for several years). EMPLOYEE_ID and PROJECT_ID in WORKS_ON are foreign keys for EMPLOYEES and PROJECTS, respectively. The three atttibutes of WORKS_ON are part of its primary key.
1a	a) Write an SQL query that allows you to obtain all employees who have worked intermittently on projects. An employee works intermittently on a project if he/she works in a year on a project (year X), does not work in a year after year X (year Y) and returns to work on that project in a year after year Y (year Z). For example, if João works on project 15 in 2014, 2015 and 2020 then we say that João worked intermittently on project 15. Another example: Maria worked on project 17 in the years 2010, 2011 and 2012 and in no other year. In this case we say that Maria worked continuously on project 17. Return the employee's name and project title. Resulting Relation: R (name, title).
1b	b) List the years that had projects going on and make sure there are no duplicate values. Resulting Relation: R (year).
1c	c) What is the highest budget of a project? Give the result the name "Highest_budget".
1d	d) Write an SQL query that returns the employees who have worked in more than 2 projects. Return the ID's of the employees and the number of projects they have worked on. Resulting Relation: R(EMPLOYEE_ID, NR_PROJECTS).
1e	e) Select the project(s) with the highest amount spent on employees' salary in 2018. Return the project title and the amount as total_salary2018. Resulting Relation: R (title, total_salary).
1f	f) Every year each project has its own costs in salaries. What was the maximum total amount spent in salaries in 2017 in a single project? Give the result the name "maximum_total_salary2017".
1g	g) Due to the current economic crises the company is no longer able to keep people with a salary higher than 2500$ working in projects this year (2021). Consequently, write a SQL query to delete from table WORKS_ON the corresponding records.
1h	h) Assume a new employee was hired. His name is 'Joaquim Mendes', he will have a salary of 1400$. Insert the new employee's data in the EMPLOYEES table. Note that the ID is a serial.

1i	i) All the employees who receive a salary below 1000$ will get a raise of 10%. Write a SQL query to update the table Employees according to this information.
1j	j) The salary of employees receiving less than 1000$ will get a raise of 10%, while employees with a salary between 1000$ and 1500$ (inclusive) will get a raise of 6%. All other employees will keep the same salary. Write a SQL query to update the table Employees accordingly.
1k	k) Return the top 5% of projects with highest budget. Resulting Relation: R (title, budget).
1l	l) Write a SQL query that returns the Projects which budget is not known. Resulting Relation: R (PROJECT_ID, title).
1m	m) There is a new employee in the company, but her salary is not defined yet. Insert the available information in the table EMPLOYEES. Her name is 'Magda Silva'. Note that the ID is a serial.
1n	n) Write an SQL query that allows you to obtain all employees whose salary is higher than 2000$ and who have worked intermittently on projects. An employee works intermittently on a project if he/she works in a year on a project (year X), does not work in a year after year X (year Y) and returns to work on that project in a year after year Y (year Z). For example, if João works on project 15 in 2014, 2015 and 2020 then we say that João worked intermittently on project 15. Another example: Maria worked on project 17 in the years 2010, 2011 and 2012 and in no other year. In this case we say that Maria worked continuously on project 17. Return the employee's name and project title. Resulting Relation: R (name, title).
2	ex002. Consider the three relations below: BOOKS(ISBN, Author, Title, Publisher, PublishDate, Pages, Notes) STORE(StoreID, StoreName, Street, State, City, Zip) STOCK(ISBN, StoreID, Price, Quantity)
2a	a) Find the name and address of the bookstore(s) carrying the smallest total quantity of books by "Fernando Pessoa", including only stores that have registered books from "Fernando Pessoa", even if Quantity=0. Resulting Relation: R (StoreName, Street, State, City, Zip).
2b	b) Find the average price of books by "Addison-Wesley" publishers in the bookstore with id 111. Give the result the name 'Addison_avg_price'.
2c	c) Find the Store Id and Store Name of all bookstores that are missing at least ten books from the BOOKS relation in their stock (for these books, either the Quantity=0 or there is no stock tuple). Resulting Relation: R (StoreID, StoreName).
2d	d) Write an SQL query that returns the ISBN of the books with a quantity in stock higher or equal to 10. Resulting Relation: R (ISBN).
2e	e) Find the titles of the books in stock (Quantity higher than 0) in both stores "Bertrand" and "Wook". Resulting Relation: R (Title).
2f	f) Find the titles of the books in stock (Quantity higher than 0) either in store "Bertrand" or "Wook". Resulting Relation: R (Title).
2g	g) Find the titles of the books in stock (Quantity higher than 0) either in store "Bertrand" or "Wook", keeping the duplicates in the case the book is available in both stores. Resulting Relation: R (Title).
2h	h) Find the titles of the books that are in stock (Quantity higher than 0) in store "Bertrand" but not in store "Wook". Resulting Relation: R (Title).
2i	i) Considering all stores, what is the total value in stock? Give the result the name "Total_value_stock".

2j	j) What is the total number of pages written by each author? Resulting Relation: R (Author, number_pages).
2k	k) Use an SQL expression to find how many books of "José Eduardo Agualusa" have stock (quantity higher than 0)? Give the result the name "Agualusa_available".
2l	l) In a single query create a temporary table called EDITORIAL_BOOKS with the same structure as BOOKS, containing only the books published by 'Porto Editora'.
2m	m) Write a SQL query that returns the table Stock and its rows, but when the Quantity is unkown assume it means there is no stock and replace the value by 0. Resulting Relation: R (ISBN, StoreID, Price, Quantity).
2n	n) Write a query that returns the name of the authors who have published more than 10 000 pages. Resulting Relation: R (Author).
2o	o) Write a query that returns the name of the authors whose average of pages per book is less than 200. Resulting Relation: R (Author).
2p	p) Write a query that returns the titles of books by 'Cesário Verde' and 'Florbela Espanca' which have less than 150 pages. Resulting Relation: R (Author, Title).
2q	q) Use an SQL expression to find how many books of "Isabel Alçada" have stock (quantity higher than 0)? Give the result the name "Isabel_available".
3	ex003. Consider the model: PERSON(PersonID, Name) RELATIONSHIP(PersonID1, Relation, PersonID2) PersonID is a PK of PERSON; The PK of RELATIONSHIP IS (PersonID1, PersonID2); PersonID1 and PersonID2 are FK's to PERSON. Relation can take the values 'Friend' and 'Enemy'.
3a	a) Write an SQL query that calculates, for each person, the total number of friends. Your query should result in the PersonID, the Name and the total number of friends. The query must include in the result all the people in the database (even those who have no friends). Resulting Relation: R (PersonID, Name, Number_of_Friends).
4	ex004. Consider the model: teams(TeamID, Name) games(WinningTeam, LoosingTeam) person(PersonID, Name, Salary, Rating) member(Player, Team) TeamID identifies teams; WinningTeam and LoosingTeam both reference a tuple in Teams; WinningTeam and LoosingTeam together identify a game; A person is identified by its PersonID; Player and Team in member reference person id and team id respectively; A member is identified by the Player and the Team (together).
4a	a) Write an SQL query that returns the names of players who are members of more than one team. Resulting Relation: R (PersonID, Name).
4b	b) Write a query that returns the name (s) of the team (s) whose sum of the salaries of their players is the highest among the various existing teams. Resulting Relation: R (Name).

4c	c) Write an SQL query that returns all players' names and respective salary if they would all get a raise of 35$. Resulting Relation: R (Name, NewSalary).
4d	d) Using an SQL expression list all players from the one with the highest rating to the one with the lowest. Resulting Relation: R (Name, Rating).
4e	e) Select the players whose salary is not known. Return only the players' names. Resulting Relation: R (Name).
4f	f) How many times did "Sporting Clube de Portugal" win a game? Give the result the name "Sporting_won".
4g	g) What is the value of the lowest salary? Give the result the name "Lowest_salary".
4h	h) How many players with a salary higher than 10000$ or a rating higher than 4 does each team have? Include all teams, even the ones with 0 players in these conditions. Resulting Relation: R (TeamID, TeamName, NrPlayers).
4i	i) List the games where either "Futebol Clube do Porto" or "Sport Lisboa e Benfica" lost. Return the name of the teams, not their ID's. Resulting Relation: R (WinningTeam_Name, LoosingTeam_Name).
4j	j) How many games has "Futebol Clube do Porto" won against "Vitória de Guimarães"? Give the result the name "Porto_wins_Guimaraes".
4k	k) Select all the games less the ones where "Futebol Clube de Famalicão" or "Clube Desportivo de Santa Clara" lost. Instead of team ID's, return their names. Resulting Relation: R (WinningTeam_Name, LoosingTeam_Name).
4l	l) For each team get the difference between games won and lost as "WonVSLost". Note that a team might never have lost or the opposite. Make sure these teams also appear on the result set. Resulting Relation: R (TeamID, TeamName, Won_Lost_Games).
4m	m) Write a SQL query that results in a table similar to the table person, with an additional column called "SalaryLevel". Select all rows from table person and to fill "Salary_Level" consider the following description: When the salary is below 5000 the value of this column should be "Low", when it is between 5000 and 12000 it is "Medium", when the salary is above 12000 the value should be "High". Resulting Relation: R (PersonID, Name, Salary, Rating, SalaryLevel).
4n	n) Write a SQL query to delete from table Person, the people who are not members of a team.
4o	o) In a single query create a new table called PeopleHighSalary only with the data of people whose salary is higher than the average.
4p	p) There will be a tournament and in the group phase each team plays against the others 2 times (home and away). Return all the games which will occur in the group composed by 'Futebol Clube do Porto', 'Sporting Clube de Portugal' and 'Sport Lisboa e Benfica'. Resulting Relation: R (HomeTeam, AwayTeam).
4q	q) The players receiving more than 100000€ will be dismissed due to lack of financial resources. Write an SQL query to delete from the table member all records associated to players who receive above this value.
4r	r) Sport Lisboa e Benfica is the only team capable of paying more than 100000€ of salary to a player. Write an SQL query to insert into table member the people who receive above this value, associating them to "Sport Lisboa e Benfica".

5	ex005. Consider the data model courses(department, number, semester, name) - a course is identified by department, number and semester. create table students(username, firstName, lastName) - username identifies a student. create table enrolled(username, department, number, semester, grade) - username, department, number, semester identify a tuple in enrolled. username references student and (department, number, semester) reference courses.
5a	a) Write a query that returns the semester(s) in which the students with username “jcf” and “jps” had at least one course together, indicating the number of courses that the said students had together in that semester. Resulting Relation: R (semester, Number_of_courses_together).
5b	b) Write a query that returns the first and last names of all students who did not attend any discipline in the semester “2017_1S”. Resulting Relation: R (firstName, lastName).
5c	c) What is the average grade for each course? Note that a course is identified by its number, semester and department. Resulting Relation: R (semester, department, name, average_grade).
5d	d) Write an SQL query which returns the name of a course if that name exists both in the "Spring" and "Fall" semesters. Resulting Relation: R (name).
5e	e) Write a SQL query to delete all courses that have never had anyone enrolled.
5f	f) Write an SQL query which returns the name of a course if that name exists both in the "Spring" and "Fall" semesters and in both semesters it belongs to the “Mathematics” department. Resulting Relation: R (name).
6	ex006. Consider the following model with stocks, stock percentage change of value (negative/positive) and investor portfolios stored in the owns table. stocks(symbol, name) - symbol identify a stock stock_var(symbol, delta, year, week) - symbol references stocks; (symbol, year, week) identify a stock_var owns(investor, symbol, quantity)- symbol references stocks; (investor,symbol) identify a tuple in owns
6a	a) Write a query that gives for each symbol (share) the average change (delta) for all weeks registered in the database. Resulting Relation: R (symbol, Average_Change).
6b	b) Write a query that gives the name of each investor and the total change in his/her portfolio for week 7 of 2017. The contribution of each symbol to the protfolio is delta*quantity. Resulting Relation: R (investor, Change_portfolio_week7_2017).
6c	c) Write a query that gives the name of each investor and the total change in his/her portfolio for week 7 of 2017. Return only investors with positive changes in their portfolio for week 7 of 2017. The contribution of each symbol to the portfolio is delta*quantity. Resulting Relation: R (investor, Change_portfolio_week7_2017).
6d	d) Write a query that returns the symbols (X) that in all weeks had a delta greater than the delta of all other symbols that don't appear in the same portfolios as X. Resulting Relation: R (symbol).

7	ex007. A web browser stores navigation information in a relational database with the following model: HISTORY(id, url, access_date) CACHE(url, content, size) BOOKMARK(name, url) Whenever the user visits a page, the browser inserts a record in the HISTORY table. The url attribute is the page's address (varchar (2048)) and access_date stores information about when the page was accessed (datetime). Some webpages are cached (CACHE). The content attribute stores the page's html code (text) and the size attribute corresponds to the number of bytes of the page (integer). The user can create bookmarks (BOOKMARK) by giving them unique names (name).
7a	a) Create a query with a list of all webpages (url) in cache, their size in bytes and the last access made to each url. Your query should output the columns: url, size, last_access, and be ordered by last_access in ascending order (older first).
8	ex008. In this question, you are given the following simple database of employees that work in specific departments. Each department has an inventory of items with specific quantity. EMPLOYEE(ssn, first_name, last_name, address, date_joined, supervisor_ssn) DEPARTMENT(dept_no, name, manager_ssn) WORKS_IN(employee_ssn, dept_no) - employee_ssn and dept_no are primary keys INVENTORY(dept_no, item_id, quantity) - dept_no and item_id are primary keys ITEMS(item_id, item_name, type) Foreign keys: 1. EMPLOYEE.supervisor_ssn and WORKS_IN.employee_ssn point to EMPLOYEE.ssn. 2. WORKS_IN.dept_no and INVENTORY.dept_no point to DEPARTMENT.dept_no. 3. INVENTORY.item_id points to ITEMS.item_id. 4. DEPARTMENT.manager_ssn points to EMPLOYEE.ssn.
8a	a) Find for all employees, their grand-supervisor, i.e. the supervisor of their immediate supervisor. Return the name of the employee and the name of their grand-supervisor, in this order. If somebody does not have a grand-supervisor, you should return next to their name. Resulting Relation: R (employee_first_name, employee_last_name, supervisor_first_name, supervisor_last_name).
8b	b) Find all departments that have a larger inventory than the department 'Joe Smith' is managing. The inventory is defined to be the sum of the quantity of items for this department. Return the name of the department and the total number of employees working in that department. Resulting Relation: R (name, Number_of_employees).
8c	c) How many employers with more than 10 years at the company does each department have? Compare to the current date. Resulting Relation: R (dept_no, dept_name, nr_employees).
8d	d) All the items of type 'x' were just sold, so the quantity in inventory is now 0. Write a SQL query to update the inventory according to this information.
8e	e) Return the top 6 items with lowest quantity in inventory. Resulting Relation: R (item_id, item_name, quantity).
8f	f) Write an SQL query that returns the full name of all employees as 'full_name'.

9	ex009. Consider the model: movies(movieID, title, premiereDate, rating int, evolution) - movieID identify a movie; evolution can only assume the values ('Positive', 'Negative', 'Stable') person(personID, name, yearBirth) - personID identify a person roles(roleID, role) - roleID identify a role; role can only assume the values ('Actor', 'Director') part(movieID, personID, roleID) - movieID references movies; personID references person; roleID references roles; (movieID, personID, roleID) identify a part.
9a	a) Taking into account the relations provided, write in SQL language a query that allows the discovery of all the directors whose films have an average rating equal to or higher than 4. Return the Person's Id, Person's Name and their Average Rating. Resulting Relation: R (personID, name, Average_Rating).
9b	b) Taking into account the relations provided, write in SQL a query that returns the movie(s) that had the largest number of actors in the cast. If the maximum number of actors in a film is twenty, for example, and there is more than one film with twenty actors in the cast, all films in which this happens must be returned. The query should return the movie id, the name of the movie and the number of actors who participated. Resulting Relation: R (movieID, title, Number_of_actors).
9c	c) Write an SQL query that returns the average rating for each value of "evolution"? Resulting Relation: R (evolution, average_rating).
9d	d) List the name of the people who were Directors of a movie before they were 30 years old. Make sure there are no duplicate values. Resulting Relation: R (name).
9e	e) Create a new table movies_simple with all the records from movies, but only 3 attributes: R(movieID, title, rating).
10	ex010. Consider a database that is intended to store information on the various cocktails produced at a bar which contain the following tables: • Cocktails (CocktailName, Price) • Ingredients (IngredientName, UnitCost, AlcoholPercentage) • Recipes (Cocktail, Ingredient, Units) The cocktail and ingredient attributes in the Recipes table are foreign keys to the Cocktails and Ingredients tables, respectively. Units represents the amount of each ingredient that goes into the composition of a cocktail.
10a	a) Using an SQL expression list the names of ingredients that cost less than $1 or are not used in any recipe. Resulting Relation: R (IngredientName)
10b	b) Using an SQL expression list the ingredient name pairs that are used in the same cocktail. Resulting Relation: R (IngredientName1, IngredientName2, CocktailName)
10c	c) Using an SQL expression list the names of the cocktails and the profit from selling them. The profit of each cocktail is calculated as the difference between its price and the cost of the ingredients used to make it. Resulting Relation: R (CocktailName, Profit)
10d	d) List the name of the cocktails which contain the ingredient "Rum". Resulting Relation: R (Cocktail).
10e	e) List all ingredients' names in alphabetical order. Resulting Relation: R (IngredientName).
10f	f) What is the average price of a cocktail? Give the result the name "Average_cocktail_price".
10g	g) List the name of the cocktails which have 4 or more different ingredients. Resulting Relation: R (Cocktail).

10h	h) Using an SQL expression select the name of the cocktails which have "Rum" and "Lemon". Resulting Relation: R (Cocktail).
10i	i) Using an SQL expression select the name of the cocktails which have "Rum" and "Lemon", or "Vodka" instead. Resulting Relation: R (Cocktail).
10j	j) Using an SQL expression select the name of the cocktails which have "Rum", but do not have "Lemon". Resulting Relation: R (Cocktail).
10k	k) List the name of the ingredient(s) with the highest alcohol percentage. Resulting Relation: R (IngredientName).
10l	l) List the name of the ingredient(s) with the lowest alcohol percentage. Resulting Relation: R (IngredientName).
10m	m) There was an update in the price of the 'Mojito' cocktail. The new price is 7$. Write an SQL query to correct the information in the Cocktails table.
10n	n) Using a SQL expression list the top 5 ingredients with highest alcohol percentage. Resulting Relation: R (IngredientName, AlcoolPercentage).
11	ex011. The following relational model is used in the Transit Authority and contains information about the citizens living in the country (PEOPLE) - identified by their citizen card (citizenCard) - and the motor vehicles in circulation (CARS) - identified by their PLATE. The model also contains a relationship with information about the owners of each vehicle (OWNS) - it is assumed that each vehicle has only one owner and each person can have 0 to many vehicles. The model also stores information about parking tickets passed to vehicles (FINES) - each fine has a unique identifier (FINEID) and, among other attributes, an attribute that saves the postcode where the fine was issued (POSTCODEFINE). PEOPLE(CITIZENCARD, NAME, PHONE, ADDRESS, POSTCODE) CARS(PLATE, COLOR, BRAND, MODEL, YEARPRODUCTION) OWNS(PLATE, CITIZENCARD) FINES(FINEID, PLATE, DATE, POSTCODEFINE)
11a	a) Obtain the total number of fines passed in 2014 for each of the postcodes of the fine. Resulting Relation: R (POSTCODEFINE, Total_fines_by_postal_code2014).
11b	b) Obtain the citizen card number and name for all persons who have received the maximum number of fines passed to one person. The result can contain more than one person. Resulting Relation: R (CITIZENCARD, NAME).
11c	c) Obtain a listing of the citizencards of all persons who have been fined in their area of residence (i.e. in their postcode of residence) and who have not been fined in areas outside their area of residence. Resulting Relation: R (CITIZENCARD).
11d	d) Find the cars of the brand "Renault" produced after 2015. Return only the cars' plates. Resulting Relation: R (PLATE).
11e	e) Write a query that returns all cars' plates and the name of the owner. Resulting Relation: R (PLATE, NAME).
11f	f) Find the cars produced between 2015 and 2017 (including these 2 years). Return only the cars' plates. Resulting Relation: R (PLATE).
11g	g) Write an SQL query that returns the total number of people who had fines in 2018. Give the result the name "People_with_fines2018".
11h	h) Select the citizencard(s) and name(s) of the citizen(s) who has/have the most recent car(s). Resulting Relation: R (CITIZENCARD, NAME).
11i	Select the name(s) of the citizen(s) who has/have the oldest car(s). Resulting Relation: R (CITIZENCARD, NAME).

11j	j) How many cars produced after 2015 does each person have? The query must include in the result all the people in the database (even those who have no cars with this condition). Resulting Relation: R (CITIZENCARD, NAME, CARS_AFTER2015).
11k	k) Write a SQL query to delete the fines which have more than 2 years old. Compare to the current date.
11l	l) Write a SQL query to delete all records on fines.
11m	m) A new car was produced. Insert its data in the Cars table. It is a 'Renault', model 'Clio', it is 'Red' and was produced in 2021. The plate is '28-IS-21'.
11n	n) List the plates of 'Red' and 'Blue' cars from 'Renault', produced before 2008. Resulting Relation: R (PLATE).
11o	o) A new car was produced, but there is no information on its color. Insert the known data in the Cars table. It is a 'Ford', model 'Fiesta' and was produced in 2021. The plate is '23-ES-21'.
12	ex012. Consider a database of an image tagging site: Members (MemberID, Name, Age) Images (ImageID, year) Tags (MemberID, ImageID) The Members relation contains one record for each member registered on the site; Images contains the metadata of the images - ImageID (PK) and year the picture was taken; Tags keep information about users who have been tagged in images. Primary keys are underlined. Non-Key Attributes May Have Values.
12a	a) Write an SQL query that returns the ID and name of all members that were tagged in images from year 2011 and year 2014 (i.e. If Pedro was only tagged in images from 2011 his ID and name should not be returned, but if he was tagged in images from 2011 and 2014 then his information should be in the result set). Return the result ordered by MemberID. Resulting Relation: R (MemberID, Name).
12b	b) Write an SQL query that returns, for each member, the total number of times that he/she has been tagged in 2015 images. Sort the results alphabetically by name. You should ensure that you include in the results all members, even those not tagged in that year. Resulting Relation: R (Name, Images_2015).
12c	c) List the name of all the members of the image tagging site. Resulting Relation: R (Name).
12d	d) How many images are stored in the database? Give the result the name "Nr_images_stored".
12e	e) How many tags does each member have? Return the member id's and respective values. Resulting Relation: R (MemberID, Nr_tags).
12f	f) Assume there are 2 new members in the image tagging website. Their id's are 138 and 139, their names are 'Fernando Silva' and 'Ana Filipe' and they are 17 and 20 years old, respectively. In a single query insert their data in the table Members.
13	ex013. Consider the following relational schema for an airline database: FLIGHTS(FlightNumber, CityOfOrigin, CityOfDestination) DEPARTURES(FlightNumber, Date, TypeOfAirplane) PASSENGERS(IDPassenger, Name, Address) RESERVATIONS(IDPassenger, FlightNumber, Date, Seat Number)
13a	Return all cities that have direct flights to Madrid and Lisboa. Resulting Relation: R (CityOfOrigin).

13b	b) Show the flight number and the date of all departures for which there are no reservations. Resulting Relation: R (FlightNumber, Date).
13c	c) Select all data from the Flights table. Resulting Relation: R (FlightNumber, CityOfOrigin, CityOfDestination).
13d	d) Select all data from DEPARTURES where the type of the airplane contains the substring "A320". This substring, if exists, is in the beginning of the expression. Resulting Relation: R (FlightNumber, Date, TypeOfAirplane).
13e	e) Select all data from DEPARTURES that happened before 2020. Resulting Relation: R (FlightNumber, Date, TypeOfAirplane).
13f	f) From all flights in table FLIGHTS, how many have origin in Lisbon? Give the result the name "Flights_from_lisbon".
13g	g) There was an update on the date of departure of flight number 19. It will depart 2 hours later than expected. Update the the table DEPARTURES accordingly.
14	ex014. In the context of a database for the Society for Animal Protection (SPA) with information regarding the adoption of domestic animals, the following physical model was produced. animalTypes(animalTypeID, animalType) - animalTypeID identify an animal type persons(personID, name, phoneNumber, address,numAnimals) - personID identify a person animals(animalID, animalTypeID, name, previousOwnerID, admissionDate) - animalID identify an animal; animalTypeID references an animal type; previousOwnerID adoption(adoptorID, animalID, adoptionDate, numChip) - adoptorID references persons; animalID references animals; (adoptorID, animalID) identifies an adoption
14a	a) Write an SQL command that allows you to calculate the total number of cats admitted to the SPA during 2015. Resulting Relation: R (Number_cats_admitted).
14b	b) Write an SQL command that lists the ID and names of the adopters who adopted at least one animal of each of the existing types. Resulting Relation: R (personID, name).
14c	c) Write an SQL command that lists the ID's and addresses of all adopters who have made at least two adoptions. Resulting Relation: R (personID, address).
14d	d) How many types of animals does each adoptor have? Resulting Relation: R (adoptorID, nr_animal_types).
14e	e) How many animals had each person adopted before 2020? Make sure all people are listed, even the ones with no adoptions before 2019. Resulting Relation: R (person_name, animals_adopted2020).
14f	f) Write an SQL query that returns the number of animals each person adopted before 2020 and since 2020 (including both years), to allow the comparison between the pre-covid and post-covid periods. Make sure all people are listed, even the ones who made no adoptions in both periods. Resulting Relation: R (name, animals_precovid, animals_postcovid).

	ex015. Consider the relational model presented below which is a very simplified version of a database of crimes occurring at several locations (e.g. “Mosteiro dos Jerónimos”, “Torre de Belém”, “Museu da Electricidade”, “City Park ”). In each location there may be several crimes and for each crime there may be several different reports. A report always has a witness and a suspect associated.
15	locations(locationID, name, city) - locationID identify a location crimes(crimeID, crimeLocation, crimeTime, crimeType, victimName) - crimeID identify a crime; crimeLocation references a location persons(personID, name) - personID identify a person reports(reportID, wittness, time, suspect, crimeID) - reportID identify a report; wittness references a person; suspect references a person; crimeID references a crime; time is a datetime value
15a	Write a SQL query that returns the names of suspected murderers (“murder” is one of the types of crimes) that occurred in the Location "Mosteiro dos Jerónimos". Resulting Relation: R (suspect).
15b	b) Write a query that returns pairs of names A, B, where A figure as witness indicating B as suspect and B figure as witness indicating A as suspect for the same crime. For example John may suspect that Peter murdered Alice and at the same time Peter suspects that John is the perpetrator of the same crime. Resulting Relation: R (witness, suspect).
15c	c) Write a query that returns for each city and type of crime the total number of crimes (Number of Crimes) and the total number of crimes solved or about to be solved (Number of Crimes Solved): a crime is considered solved or about to be solved whenever there is at least one report of this crime and all existing reports indicate the same suspect. You may have to write several SQL statements to get this answer by storing the resulting information from the intermediate steps. Resulting Relation: R (city, crimeType, Number_of_Crimes, Number_Crimes_Solved).
15d	d) Write an SQL expression that returns the day/time, the type of crime and the name of the victim for all crimes committed in Lisbon. Resulting Relation: R (crimeTime, crimeType, victimName).
15e	e) Write an SQL expression that returns cities where no crime was committed. Resulting Relation: R (city).
15f	f) Find the types of crime that have happened in "Lisbon". Make sure there are no duplicate values. Resulting Relation: R (crimeType).
15g	g) How many people are suspects of a crime? Give the result the name "Nr_people_suspects".
15h	h) How many cities have had crimes of "Homicide"? Give the result the name “Cities_with_homicides".
15i	Write a query that returns the CrimeID and Victim Name of all crimes of "murder" and "sexual assault" that occurred in locations in "Lisbon". Resulting Relation: R (crimeID, victimName).
15j	j) Write a query that returns the first wittness to report the crime with ID number 2. Assume it is impossible that multiple reports from the same crime are submitted at the same time. Resulting Relation: R (personID, name).
15k	k) Write an SQL query that returns the number of hours that went by since the first report of crimeID 2 was submitted until the last report of this crimeID was submitted. Give the result the name "Time_Between".
15l	l) Write a query that returns the CrimeID and Victim Name of all crimes of "theft" and "burglary" that occurred in locations in "Porto". Resulting Relation: R (crimeID, victimName).

ex016.

Consider the following relational schema for a database storing information on persons' appointments in citizen shops:

16

Person(Person_ID, Name, Email, Date_Of_Birth) - Person_ID identify a person
Appointments(Appointment_ID, Person_ID, Shop_ID, Datetime) - Appointment_ID identify an appointment; Person_ID references a person; Shop_ID references a citizen shop
Citizen_Shops(Shop_ID, Name, Address, Phone_Number) - Shop_ID identify a citizen shop

16a

a) Write a SQL query that returns the ID and name of all people with scheduled appointments in the future.
Each person might have multiple appointments scheduled, make sure she/he only appears once in the result set. Note that the Appointments table also stores past data.
Resulting Relation: R (PersonID, Name).

16b

b) The opening and closing hours for the Citizen Shop with ID 2 have changed and it will now open one hour later and close also one hour later. Write a SQL query that returns all the appointments scheduled for the future in the citizen shop with ID 2 and add one hour to the date of all those appointments. Resulting Relation: R (PersonID, New_Date).

16c

c) Strangely there is a person with a date of birth posterior to the date of that person's appointment (she/he only has one appointment scheduled).
Write a query that returns this person ID, name, date of birth and the date of the appointment.
Resulting Relation: R (PersonID, Name, Date_Of_Birth, Date_Of_Appointment).

Table 11 - Identified Tasks (Multiple-Choice)

ID	Question Statement
17	Suppose that R (a,b) contains the tuples {(1,2), (3,4)} and that S(b,c) contains the tuples {(1,2), (1,3), (2,3), (2,1)}. Which of the following is **true**? a) The statement SELECT * FROM R FULL OUTER JOIN S ON R.B = S.B returns 5 tuples b) The expression SELECT * FROM R INNER JOIN S ON A<C returns 2 tuples c) The **cartesian product** of the relations R and S has the tuple (1,2,3) d) The **left outer join** of the relations R and S has the tuple (null,null,1,2)
18	Given the tables R(A,B) and S(B,C) with the tuples: R(A,B): {(30,1), (31,2), (62,3), (77,5), (19,8)} S(B,C): {(2,600), (3,520), (6,50), (6,150), (7,600)} The table obtained from the statement **SELECT * FROM R FULL OUTER JOIN S ON R.B = S.B** has: a) 2 tuples b) 25 tuples c) 8 tuples d) None of the alternatives is correct
19	Which of the following statements is **true**? a) The attributes used in the Group By clause must be present in the Select statement b) When a Group By clause is used, the Select list can only have the attributes used in the Group By and aggregation functions c) When a Group By clause is used, the Select list can only have the attributes used in the Group By, the attributes used in the JOIN (if there is one) and aggregation functions d) When a Group By clause is used, the Select list is entirely composed by aggregation functions
20	Which of the following statements is **true**? a) When both Group By and Where clauses exist in a query, the conditions in the Where are applied to the grouped values b) When both Group By and Having clauses exist in a query, the conditions in the Having are applied to the grouped values c) The Having clause must be used when a Group By clause exists d) The Having clause may be used without a Group By clause
21	Consider relations R(a,b) and S(c,d), where a is the primary key of R, c is the primary key of S and b is a foreign key of R, referencing c. R(a,b) contains 4 tuples: (0,4), (1,5), (2,4), (3,5) S(c,d) also contains 4 tuples: (2,10), (3,11), (4,12), (5,13) Which of the following modifications **will not be rejected** due to a violation of a constraint? a) DELETE of S tuple (3,11) b) INSERT of tuple (4,4) in S c) INSERT of S tuple (3,3) d) DELETE S tuple (4,12)

Consider relations R(a,b) and S(c,d), where a is the primary key of R, c is the primary key of S and b is a foreign key of R, referencing c.
R(a,b) contains 4 tuples: (0,4), (1,5), (2,4), (3,5)
S(c,d) also contains 4 tuples: (2,10), (3,11), (4,12), (5,13)

22 Which of the following modifications **will not be rejected** due to a violation of a constraint?

a) INSERT of S tuple (4,13)
b) INSERT of R tuple (5,0)
c) UPDATE of R tuple (0,4) changing to (0,0)
d) INSERT of R tuple (5,2)

Consider the R relation (a,b,c) where a is the primary key.
R has the tuple (1,2,3). Which of the following inserts **will not be rejected** due to a violation of a constraint?

23

a) INSERT INTO R(a,b,c) VALUES (1,4,7);
b) INSERT INTO R(b,c) VALUES (3,4);
c) INSERT INTO R(c,b) VALUES (3,4);
d) INSERT INTO R(a,b,c) VALUES (2,2,3);

Given the tables R(A,B) and S(B,C) with the tuples:
R(A,B): {(30,1), (31,2), (62,3), (77,5), (19,8)}
S(B,C): {(2,600), (3,520), (6,50), (6,150), (7,600)}
The table obtained from the statement

24 **SELECT * FROM R LEFT OUTER JOIN S ON R.B = S.B**

a) 8 tuples
b) 25 tuples
c) 2 tuples
d) None of the alternatives is correct

From the following statements select the one (only one) that is **incorrect**

a) A relation can only have one primary key and one alternate key

25

b) A relation must have an alternate key
c) A primary key cannot assume NULL values
d) An alternate key can assume NULL values

X and Y are two relations such that X has 1 column and 2 tuples and Y has 3 columns and 4 tuples. How many tuples result from a CROSS JOIN between X and Y?

26

a) 4
b) 6
c) 8
d) None of the previous options

Given two relations R1(A,B) and R2(A,B), where R1 contains N1 tuples and R2 contains N2 tuples and N2 > N1 > 0, say which of the following statements is **true**.

a) The minimum and maximum number of tuples that are obtained from the UNION of the tuples in both tables is N2 and N1*N2 respectively

27

b) The minimum and maximum number of tuples that are obtained from the UNION of the tuples in both tables is N2 and N1+N2 respectively
c) The minimum and maximum number of tuples that are obtained from the INTERSECTION of the tuples in both tables is 0 and N2 respectively
d) The minimum and maximum number of tuples that are obtained from the INTERSECTION of the tuples in both tables is N1 and N2 respectively

---

28

Given two relations R1(A,B) and R2(A,B), where R1 contains N1 tuples and R2 contains N2 tuples and N2 > N1 > 0, say which of the following statements is **true**.

a) The minimum and maximum number of tuples that are obtained when doing the tuples of R1 EXCEPT the tuples of R2 is 0 and N1 respectively
b) The minimum and maximum number of tuples that are obtained from the CROSS JOIN of the tuples of R1 and R2 is N1 and N2 respectively
c) The minimum and maximum number of tuples that are obtained when doing the tuples of R1 EXCEPT the tuples of R2 is N1 and N2 respectively
d) The minimum and maximum number of tuples that are obtained from the INTERSECTION of the tuples in R1 and R2 is N1 and N2 respectively

29

What does UNION operator do in a SQL Server statement?

a) Returns all data from the listed tables
b) Returns all distinct data from the listed tables
c) Returns the common data from the listed tables
d) Returns data which exists in one of the listed tables but not the other

30

What does UNION ALL operator do in a SQL Server statement?

a) Returns all data from the listed tables
b) Returns all distinct data from the listed tables
c) Returns the common data from the listed tables
d) Returns data which exists in one of the listed tables but not the other

31

What does EXCEPT operator do in a SQL Server statement?

a) Returns all data from the listed tables
b) Returns all distinct data from the listed tables
c) Returns the common data from the listed tables
d) Returns data which exists in one of the listed tables but not the other

32

What does INTERSECT operator do in a SQL Server statement?

a) Returns all data from the listed tables
b) Returns all distinct data from the listed tables
c) Returns the common data from the listed tables
d) Returns data which exists in one of the listed tables but not the other

33

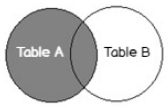
A Composite Primary Key is

a) A primary key with int as the field type
b) Is the same as a foreign key
c) A primary key made up of two or more fields
d) None of the above

34

Consider the image below and choose the type of join it illustrates

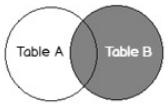
a) Left Join
b) Inner Join
c) Right Join
d) Full Outer Join



35

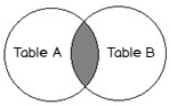
Consider the image below and choose the type of join it illustrates

a) Left Join
b) Inner Join
c) Right Join
d) Full Outer Join



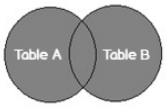
36 Consider the image below and choose the type of join it illustrates

a) Left Join
b) Inner Join
c) Right Join
d) Full Outer Join



37 Consider the image below and choose the type of join it illustrates

a) Left Join
b) Inner Join
c) Right Join
d) Full Outer Join



38 Which join is equivalent to Cartesian Product?

a) Inner Join
b) Cross Join
c) Full Outer Join
d) Self Join

39 Which of the following statements is **incorrect**?

a) In the result of a FULL OUTER JOIN between tables A and B, the key column from table A and the key column from table B can both assume null values
b) In the result of a LEFT JOIN between tables A and B, the key column from table A cannot assume null values but the key column from table B can
c) In the result of an INNER JOIN between tables A and B, the key column from table A and the key column from table B cannot assume null values
d) In the result of a LEFT JOIN between tables A and B, the key column from table A can assume null values but the key column from table B cannot

40 Choose the right expression for the space in blank so that the below statement is **correct**. SELF JOIN can be attained by _____ based on the requirement(s) but the table must join with itself.

a) An Inner join or a Left/Right Join
b) An Inner Join or a Full Outer Join
c) A Full Outer Join or a Left/Right Join
d) None of the above

41 Select the SQL operator which performs the operation stated in the sentence below. "Compare values from a column with values resulting from a subquery. Return true if the operation is true for any of the values in the subquery result."

a) LIKE
b) ANY
c) ALL
d) DISTINCT

42 Select the SQL operator which performs the operation stated in the sentence below. "Compare values from a column with values resulting from a subquery. Return true if the operation is true for all the values in the subquery result."

a) LIKE
b) ANY
c) ALL
d) DISTINCT

10. APPENDIX C

Table 12 - Task-Competency Matrix

TaskID / CompetencyID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	Minimal Subset of Competencies
1a_1	X		X	X	X	X		X											X																	X	X				X	X			X													X																				[1, 3, 4, 5, 6, 8, 19, 36, 37, 41, 42, 45, 58]
1a_2	X		X	X	X	X													X		X	X					X									X	X																	X	X	X	X																						[1, 3, 4, 5, 6, 19, 21, 22, 27, 36, 37, 54, 55, 56, 57]
1b	X		X																																																																											[1, 3]
1c																	X	X			X																																																									[17, 18, 21]
1d	X				X												X	X								X																												X	X	X	X																						[1, 5, 17, 18, 26, 54, 55, 56, 57]
1e	X			X	X												X	X						X									X	X		X	X																	X	X					X	X																		[1, 4, 5, 17, 18, 24, 33, 34, 36, 37, 54, 55, 60, 61]
1f	X			X	X												X	X			X			X									X	X		X	X																	X	X																								[1, 4, 5, 17, 18, 21, 24, 33, 34, 36, 37, 54, 55]
1g				X	X	X																														X	X																																						X					[4, 5, 6, 36, 37, 73]
1h																																																																				X											[68]
1i				X	X														X																																																			X								[4, 5, 19, 70]
1j					X	X													X																																													X						X								[5, 6, 19, 64, 70]
1k	X										X		X																																																																	[1, 11, 13]
1l	X			X																																																										X																[1, 4, 62]
1m																																																																				X											[68]
1n_1	X		X	X	X	X		X											X																	X	X				X	X			X													X																				[1, 3, 4, 5, 6, 8, 19, 36, 37, 41, 42, 45, 58]
1n_2	X		X	X	X	X													X		X	X					X									X	X																	X	X	X	X																						[1, 3, 4, 5, 6, 19, 21, 22, 27, 36, 37, 54, 55, 56, 57]
2a	X			X	X													X						X									X			X	X																	X	X	X	X			X	X																		[1, 4, 5, 18, 24, 33, 36, 37, 54, 55, 56, 57, 60, 61]
2b				X	X	X											X	X					X													X	X																																										[4, 5, 6, 17, 18, 23, 36, 37]
2c	X			X	X													X	X							X	X								X	X	X																	X	X	X	X																						[1, 4, 5, 18, 19, 26, 27, 35, 36, 37, 54, 55, 56, 57]
2d	X			X	X																															X	X																																										[1, 4, 5, 36, 37]
2e	X			X	X	X																														X	X													X	X																												[1, 4, 5, 6, 36, 37, 50, 51]
2f	X			X	X	X																														X	X									X	X																														[1, 4, 5, 6, 36, 37, 46, 47]
2g	X			X	X	X																														X	X											X	X																												[1, 4, 5, 6, 36, 37, 48, 49]
2h	X			X	X	X																														X	X															X	X																									[1, 4, 5, 6, 36, 37, 52, 53]
2i																	X	X	X					X																																																						[17, 18, 19, 24]
2j	X																X	X						X																														X	X																								[1, 17, 18, 24, 54, 55]
2k_1				X	X	X											X	X								X										X	X																																										[4, 5, 6, 17, 18, 26, 36, 37]
2k_2				X	X	X										X	X	X							X																																																						[4, 5, 6, 16, 17, 18, 25]
2l		X		X	X																																																													X												[2, 4, 5, 66]
2m	X																X	X																																													X															[1, 17, 18, 63]
2n	X				X																			X																														X	X	X	X																						[1, 5, 24, 54, 55, 56, 57]
2o	X				X																		X																															X	X	X	X																						[1, 5, 23, 54, 55, 56, 57]
2p_1	X			X	X	X									X																																																															[1, 4, 5, 6, 15]
2p_2	X			X	X	X	X		X																																																																						[1, 4, 5, 6, 7, 9]
2q_1				X	X	X										X	X	X							X																																																						[4, 5, 6, 16, 17, 18, 25]
2q_2				X	X	X											X	X								X										X	X																																										[4, 5, 6, 17, 18, 26, 36, 37]
3a	X			X	X		X										X	X									X											X	X															X	X																								[1, 4, 5, 7, 17, 18, 27, 38, 39, 54, 55]
4a	X				X																						X									X	X																	X	X	X	X																						[1, 5, 27, 36, 37, 54, 55, 56, 57]
4b	X				X													X						X												X	X																	X	X	X	X			X	X																		[1, 5, 18, 24, 36, 37, 54, 55, 56, 57, 60, 61]
4c	X																X	X	X																																																												[1, 17, 18, 19]
4d	X										X																																																																				[1, 11]
4e	X			X																																																									X																[1, 4, 62]
4f				X	X												X	X									X									X	X																																										[4, 5, 17, 18, 27, 36, 37]
4g																	X	X				X																																																								[17, 18, 22]
4h	X			X	X		X										X	X							X													X	X															X	X																								[1, 4, 5, 7, 17, 18, 25, 38, 39, 54, 55]
4i	X			X	X		X										X																X			X	X				X	X																																							[1, 4, 5, 7, 17, 33, 36, 37, 41, 42]
4j				X	X	X											X	X									X						X			X	X				X	X																																							[4, 5, 6, 17, 18, 27, 33, 36, 37, 41, 42]
4k	X			X	X			X							X		X																			X	X				X	X																																							[1, 4, 5, 8, 15, 17, 36, 37, 41, 42]
4l	X				X												X	X	X								X						X	X				X	X															X	X																								[1, 5, 17, 18, 19, 27, 33, 34, 38, 39, 54, 55]
4m	X				X	X											X	X																																														X														[1, 5, 6, 17, 18, 64]
4n	X			X				X								X																																																								X						[1, 4, 8, 16, 72]
4o		X		X	X													X					X												X																														X													[2, 4, 5, 18, 23, 35, 65]
4p	X			X	X	X									X		X																X			X	X				X	X																																							[1, 4, 5, 6, 15, 17, 33, 36, 37, 41, 42]
4q				X	X	X																														X	X																																						X					[4, 5, 6, 36, 37, 73]
4r	X			X	X																																							X																										X									[1, 4, 5, 44, 69]
5a	X			X	X	X											X	X									X						X			X	X				X	X												X	X																								[1, 4, 5, 6, 17, 18, 27, 33, 36, 37, 41, 42, 54, 55]
5b	X			X	X			X								X																																																														[1, 4, 5, 8, 16]
5c	X				X	X											X	X					X													X	X																	X	X																								[1, 5, 6, 17, 18, 23, 36, 37, 54, 55]
5d_1	X			X	X	X																											X												X													X																				[1, 4, 5, 6, 33, 45, 58]
5d_2	X			X	X																																													X	X																											[1, 4, 5, 50, 51]
5e		X		X	X	X		X																																					X													X														X						[2, 4, 5, 6, 8, 45, 58, 72]
5f	X			X	X	X																											X												X													X																				[1, 4, 5, 6, 33, 45, 58]
6a	X																X	X					X																															X	X																								[1, 17, 18, 23, 54, 55]
6b	X			X	X	X											X	X	X					X												X	X																	X	X																								[1, 4, 5, 6, 17, 18, 19, 24, 36, 37, 54, 55]
6c	X			X	X	X											X	X	X					X												X	X																	X	X	X	X																						[1, 4, 5, 6, 17, 18, 19, 24, 36, 37, 54, 55, 56, 57]
6d	X			X	X	X		X																									X			X	X				X	X			X													X																				[1, 4, 5, 6, 8, 33, 36, 37, 41, 42, 45, 58]
7a	X				X					X							X	X			X															X	X																	X	X																								[1, 5, 10, 17, 18, 21, 36, 37, 54, 55]

TaskID / CompetencyID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	Minimal Subset of Competencies
8a	X				X												X																X					X	X		X	X																																				[1, 5, 17, 33, 38, 39, 41, 42]
8b	X			X	X	X											X	X						X		X									X	X	X							X	X									X	X	X	X																					[1, 4, 5, 6, 17, 18, 24, 26, 35, 36, 37, 44, 45, 54, 55, 56, 57]
8c	X			X	X												X	X							X				X			X				X	X																	X	X																							[1, 4, 5, 17, 18, 25, 29, 32, 36, 37, 54, 55]
8d				X	X	X																														X	X																																		X							[4, 5, 6, 36, 37, 71]
8e	X				X					X		X																								X	X																																									[1, 5, 10, 12, 36, 37]
8f																X	X		X																																																										[17, 18, 20]
9a	X			X	X												X	X					X													X	X																	X	X	X	X																					[1, 4, 5, 17, 18, 23, 36, 37, 54, 55, 56, 57]
9b	X			X	X												X	X									X									X	X																	X	X	X	X			X	X																	[1, 4, 5, 17, 18, 27, 36, 37, 54, 55, 56, 57, 60, 61]
9c	X																X	X					X																															X	X																							[1, 17, 18, 23, 54, 55]
9d	X		X	X	X	X													X												X					X	X																																									[1, 3, 4, 5, 6, 19, 31, 36, 37]
9e	X																																																																	X												[1, 66]
10a	X			X	X		X	X								X																																																														[1, 4, 5, 7, 8, 16]
10b	X			X	X	X											X																X			X	X				X	X																																				[1, 4, 5, 6, 17, 33, 36, 37, 41, 42]
10c	X				X												X	X	X					X												X	X																	X	X																							[1, 5, 17, 18, 19, 24, 36, 37, 54, 55]
10d	X			X	X																																																																									[1, 4, 5]
10e	X									X																																																																				[1, 10]
10f																X	X					X																																																							[17, 18, 23]
10g	X				X																				X																													X	X	X	X																					[1, 5, 25, 54, 55, 56, 57]
10h	X			X	X																																													X	X																											[1, 4, 5, 50, 51]
10i	X			X	X																																									X	X			X	X																											[1, 4, 5, 46, 47, 50, 51]
10j	X			X	X																																															X	X																									[1, 4, 5, 52, 53]
10k	X			X	X																																																							X	X																	[1, 4, 5, 60, 61]
10l	X			X	X																																																							X	X																	[1, 4, 5, 60, 61]
10m				X	X																																																																	X								[4, 5, 70]
10n	X										X	X																																																																		[1, 11, 12]
11a	X			X	X												X	X									X				X																							X	X																							[1, 4, 5, 17, 18, 27, 31, 54, 55]
11b	X				X												X	X			X						X						X	X	X	X	X																	X	X	X	X																					[1, 5, 17, 18, 21, 27, 33, 34, 35, 36, 37, 54, 55, 56, 57]
11c	X			X	X																															X	X															X	X																									[1, 4, 5, 36, 37, 52, 53]
11d	X			X	X	X																																																																								[1, 4, 5, 6]
11e	X				X																															X	X																																									[1, 5, 36, 37]
11f	X			X	X	X																																																																								[1, 4, 5, 6]
11g				X	X												X	X								X					X					X	X																																									[4, 5, 17, 18, 26, 31, 36, 37]
11h	X		X	X	X																															X	X																							X	X																	[1, 3, 4, 5, 36, 37, 60, 61]
11i	X		X	X	X																															X	X																							X	X																	[1, 3, 4, 5, 36, 37, 60, 61]
11j	X			X	X												X	X							X													X	X															X	X																							[1, 4, 5, 17, 18, 25, 38, 39, 54, 55]
11k				X	X																								X			X																																								X						[4, 5, 29, 32, 72]
11l																																																																								X						[72]
11m																																																																			X											[67]
11n_1	X			X	X	X									X																																																															[1, 4, 5, 6, 15]
11n_2	X			X	X	X	X		X																																																																					[1, 4, 5, 6, 7, 9]
11o																																																																				X										[68]
12a	X			X	X		X			X																X										X	X																	X	X	X	X																					[1, 4, 5, 7, 10, 26, 36, 37, 54, 55, 56, 57]
12b	X			X	X					X							X	X							X													X	X															X	X																							[1, 4, 5, 10, 17, 18, 25, 38, 39, 54, 55]
12c	X																																																																													[1]
12d																X	X									X																																																			[17, 18, 27]
12e	X																X	X							X																													X	X																							[1, 17, 18, 25, 54, 55]
12f																																																																			X											[67]
13a	X		X	X	X																																													X	X																											[1, 3, 4, 5, 50, 51]
13b	X			X	X	X		X																																					X													X																				[1, 4, 5, 6, 8, 45, 58]
13c		X																																																																												[2]
13d		X		X										X																																																																[2, 4, 14]
13e		X		X	X																										X																																															[2, 4, 5, 31]
13f				X	X												X	X							X																																																					[4, 5, 17, 18, 25]
13g				X	X																									X																																								X								[4, 5, 30, 70]

TaskID / CompetencyID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	Minimal Subset of Competencies
14a				x	x	x											x	x									x				x					x	x																																									[4, 5, 6, 17, 18, 27, 31, 36, 37]
14b	x				x													x								x	x								x	x	x																	x	x	x	x																						[1, 5, 18, 26, 27, 35, 36, 37, 54, 55, 56, 57]
14c	x				x																					x						x	x		x	x																	x	x	x	x																						[1, 5, 27, 33, 34, 36, 37, 54, 55, 56, 57]
14d	x				x												x	x								x										x	x																	x	x																								[1, 5, 17, 18, 26, 36, 37, 54, 55]
14e	x			x	x												x	x							x						x							x	x														x	x																								[1, 4, 5, 17, 18, 25, 31, 38, 39, 54, 55]
14f	x			x	x												x	x							x						x		x	x		x	x	x	x															x	x																								[1, 4, 5, 17, 18, 25, 31, 33, 34, 36, 37, 38, 39, 54, 55]
15a	x		x	x	x	x																														x	x																																											[1, 3, 4, 5, 6, 36, 37]
15b	x		x	x	x	x																											x			x	x				x	x																																						[1, 3, 4, 5, 6, 33, 36, 37, 41, 42]
15c	x			x	x	x											x	x							x	x							x	x		x	x	x	x															x	x									x	x															[1, 4, 5, 6, 17, 18, 25, 26, 33, 34, 36, 37, 38, 39, 54, 55, 63, 64]
15d	x			x	x																															x	x																																										[1, 4, 5, 36, 37]
15e	x				x																															x	x															x	x																											[1, 5, 36, 37, 52, 53]
15f	x		x	x	x																															x	x																																										[1, 3, 4, 5, 36, 37]
15g																	x	x								x																																																						[17, 18, 26]
15h				x	x												x	x								x										x	x																																										[4, 5, 17, 18, 26, 36, 37]
15i_1	x			x	x	x	x		x							x																																																														[1, 4, 5, 6, 7, 9, 16]
15i_2	x			x	x	x									x	x																																																														[1, 4, 5, 6, 15, 16]
15i_3	x			x	x	x									x																					x	x																																										[1, 4, 5, 6, 15, 36, 37]
15i_4	x			x	x	x	x		x																											x	x																																										[1, 4, 5, 6, 7, 9, 36, 37]
15j	x			x	x						x	x																								x	x																																										[1, 4, 5, 11, 12, 36, 37]
15k				x	x												x	x			x	x										x																																														[4, 5, 17, 18, 21, 22, 32]
15l_1	x			x	x	x									x	x																																																														[1, 4, 5, 6, 15, 16]
15l_2	x			x	x	x	x		x							x																																																														[1, 4, 5, 6, 7, 9, 16]
16a	x		x	x	x																							x	x							x	x																																										[1, 3, 4, 5, 28, 29, 36, 37]
16b	x			x	x	x											x											x	x	x																																																[1, 4, 5, 6, 17, 28, 29, 30]
16c	x			x	x												x											x								x	x																																										[1, 4, 5, 17, 28, 36, 37]
17																																				x		x		x			x																																			[36, 38, 40, 43]
18																																								x																																						[40]
19																																																					|x|																								[54]
20																																																						x		x																							[54, 56]
21																																																																									x		x		[74, 76]
22																																																																									x		x		[74, 76]
23																																																																									x				[74]
24																																					|x|																																									[38]
25																																																																									x			x	[74, 77]
26																																										|x|																																			[43]
27																																														x				x																													[46, 50]
28																																											x							x		x																											[43, 50, 52]
29																																														x																																[46]
30																																																x																														[48]
31																																																			|x|																										[52]
32																																																	|x|																												[50]
33																																																																									x	x			[74, 75]
34																																					|x|																																								[38]
35																																					|x|																																								[38]
36																																			|x|																																										[36]
37																																								x																																						[40]
38																																										|x|																																			[43]
39																																				x		x		x																																								[36, 38, 40]
40																																				x		x			x																																						[36, 38, 41]
41																																																										|x|																			[59]
42																																																											|x|																		[60]

11. APPENDIX D

Table 13 - Base states of the Competence Space and associated Performance states

ID	Competence Base States	Performance Base states
1	{1}	{12c}
2	{2}	{13c}
3	{36}	{36}
4	{38}	{24, 34, 35}
5	{40}	{18, 37}
6	{43}	{26, 38}
7	{46}	{29}
8	{48}	{30}
9	{50}	{32}
10	{52}	{31}
11	{54}	{19}
12	{59}	{41}
13	{60}	{42}
14	{67}	{11m, 12f}
15	{68}	{11o, 1m, 1h}
16	{72}	{11l}
17	{74}	{23}
18	{1, 10}	{10e}
19	{1, 3}	{1b}
20	{1, 11}	{4d}
21	{1, 66}	{9e}
22	{54, 56}	{20}
23	{74, 76}	{21, 22}
24	{74, 77}	{25}
25	{74, 75}	{33}
26	{1, 4, 5}	{10d}
27	{1, 11, 12}	{10n}
28	{1, 11, 13}	{1k}
29	{1, 4, 62}	{4e, 1l}
30	{2, 4, 14}	{13d}
31	{4, 5, 70}	{10m}
32	{17, 18, 23}	{10f}
33	{17, 18, 27}	{12d}
34	{17, 18, 26}	{15g}
35	{17, 18, 21}	{1c}
36	{17, 18, 22}	{4g}
37	{17, 18, 20}	{8f}

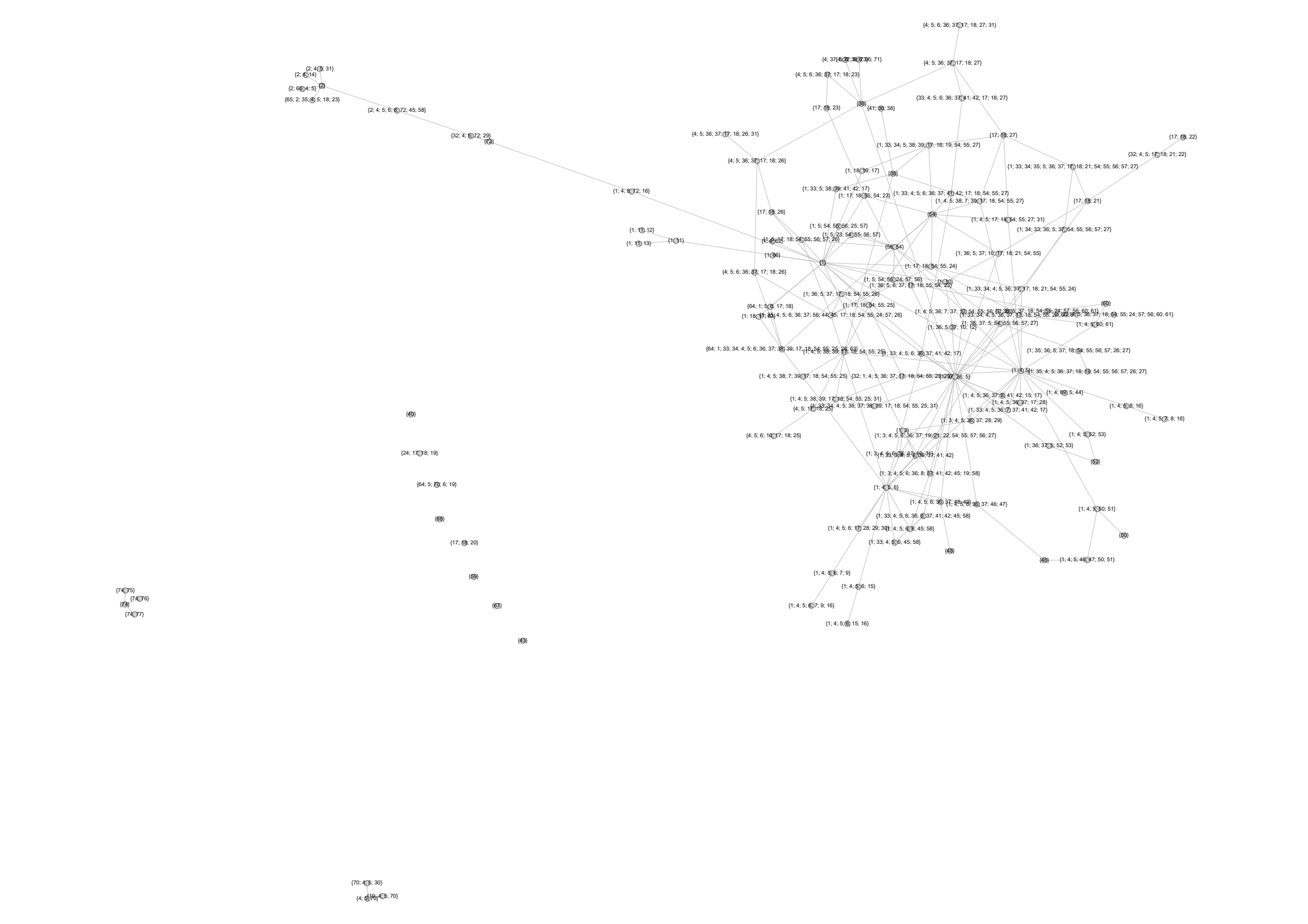
38	{36, 38, 41}	{40}
39	{1, 4, 5, 6}	{11d, 11f}
40	{1, 5, 36, 37}	{11e}
41	{1, 17, 18, 63}	{2m}
42	{1, 17, 18, 19}	{4c}
43	{2, 4, 5, 31}	{13e}
44	{2, 4, 5, 66}	{2l}
45	{4, 5, 30, 70}	{13g}
46	{4, 5, 19, 70}	{1i}
47	{17, 18, 19, 24}	{2i}
48	{1, 4, 5, 50, 51}	{10h, 5d_2}
49	{1, 4, 5, 52, 53}	{10j}
50	{1, 4, 5, 60, 61}	{10l , 10k }
51	{1, 4, 5, 6, 15}	{11n_1}
52	{1, 4, 8, 16, 72}	{4n}
53	{1, 4, 5, 44, 69}	{4r}
54	{1, 4, 5, 8, 16}	{5b}
55	{4, 5, 29, 32, 72}	{11k}
56	{4, 5, 17, 18, 25}	{13f}
57	{5, 6, 19, 64, 70}	{1j}
58	{1, 4, 5, 7, 8, 16}	{10a}
59	{1, 17, 18, 25, 54, 55}	{12e}
60	{1, 5, 36, 37, 52, 53}	{15e}
61	{1, 4, 5, 6, 15, 16}	{15i_2}
62	{1, 17, 18, 24, 54, 55}	{2j}
63	{1, 4, 5, 6, 7, 9}	{11n_2}
64	{1, 5, 6, 17, 18, 64}	{4m}
65	{1, 5, 10, 12, 36, 37}	{8e}
66	{1, 17, 18, 23, 54, 55}	{9c, 6a}
67	{4, 5, 6, 36, 37, 73}	{4q, 1g}
68	{4, 5, 6, 36, 37, 71}	{8d}
69	{1, 5, 25, 54, 55, 56, 57}	{10g}
70	{1, 4, 5, 46, 47, 50, 51}	{10i}
71	{1, 4, 5, 6, 8, 45, 58}	{13b}
72	{1, 4, 5, 6, 7, 9, 16}	{15i_1}
73	{1, 4, 5, 17, 28, 36, 37}	{16c}
74	{1, 5, 24, 54, 55, 56, 57}	{2n}
75	{1, 5, 23, 54, 55, 56, 57}	{2o}
76	{1, 4, 5, 6, 33, 45, 58}	{5d_1}
77	{2, 4, 5, 18, 23, 35, 65}	{4o}
78	{4, 5, 17, 18, 26, 36, 37}	{15h}
79	{4, 5, 17, 18, 21, 22, 32}	{15k}

80	{4, 5, 6, 16, 17, 18, 25}	{2k_2}
81	{4, 5, 17, 18, 27, 36, 37}	{4f}
82	{1, 3, 4, 5, 28, 29, 36, 37}	{16a}
83	{1, 4, 5, 6, 17, 28, 29, 30}	{16b}
84	{1, 4, 5, 6, 36, 37, 46, 47}	{2f}
85	{1, 4, 5, 6, 36, 37, 48, 49}	{2g}
86	{1, 5, 17, 33, 38, 39, 41, 42}	{8a}
87	{2, 4, 5, 6, 8, 45, 58, 72}	{5e}
88	{4, 5, 17, 18, 26, 31, 36, 37}	{11g}
89	{4, 5, 6, 17, 18, 23, 36, 37}	{2b}
90	{4, 5, 6, 17, 18, 26, 36, 37}	{2k_1}
91	{1, 4, 5, 17, 18, 27, 31, 54, 55}	{11a}
92	{1, 5, 17, 18, 26, 36, 37, 54, 55}	{14d}
93	{1, 5, 17, 18, 26, 54, 55, 56, 57}	{1d}
94	{1, 5, 27, 36, 37, 54, 55, 56, 57}	{4a}
95	{1, 3, 4, 5, 6, 19, 31, 36, 37}	{9d}
96	{4, 5, 6, 17, 18, 27, 31, 36, 37}	{14a}
97	{1, 4, 5, 6, 17, 33, 36, 37, 41, 42}	{10b}
98	{1, 4, 5, 17, 18, 25, 38, 39, 54, 55}	{11j}
99	{1, 3, 4, 5, 6, 33, 36, 37, 41, 42}	{15b}
100	{1, 4, 5, 7, 17, 33, 36, 37, 41, 42}	{4i}
101	{1, 4, 5, 8, 15, 17, 36, 37, 41, 42}	{4k}
102	{1, 5, 6, 17, 18, 23, 36, 37, 54, 55}	{5c}
103	{1, 5, 10, 17, 18, 21, 36, 37, 54, 55}	{7a}
104	{1, 5, 27, 33, 34, 36, 37, 54, 55, 56, 57}	{14c}
105	{1, 4, 5, 17, 18, 25, 31, 38, 39, 54, 55}	{14e}
106	{1, 4, 5, 7, 17, 18, 27, 38, 39, 54, 55}	{3a}
107	{1, 4, 5, 7, 17, 18, 25, 38, 39, 54, 55}	{4h}
108	{4, 5, 6, 17, 18, 27, 33, 36, 37, 41, 42}	{4j}
109	{1, 4, 5, 7, 10, 26, 36, 37, 54, 55, 56, 57}	{12a}
110	{1, 5, 18, 26, 27, 35, 36, 37, 54, 55, 56, 57}	{14b}
111	{1, 5, 18, 24, 36, 37, 54, 55, 56, 57, 60, 61}	{4b}
112	{1, 5, 17, 18, 19, 27, 33, 34, 38, 39, 54, 55}	{4l}
113	{1, 4, 5, 6, 8, 33, 36, 37, 41, 42, 45, 58}	{6d}
114	{1, 4, 5, 17, 18, 25, 29, 32, 36, 37, 54, 55}	{8c}
115	{1, 3, 4, 5, 6, 8, 19, 36, 37, 41, 42, 45, 58}	{1a_1}
116	{1, 4, 5, 17, 18, 21, 24, 33, 34, 36, 37, 54, 55}	{1f}
117	{1, 4, 5, 17, 18, 24, 33, 34, 36, 37, 54, 55, 60, 61}	{1e}
118	{1, 4, 5, 18, 24, 33, 36, 37, 54, 55, 56, 57, 60, 61}	{2a}
119	{1, 4, 5, 18, 19, 26, 27, 35, 36, 37, 54, 55, 56, 57}	{2c}
120	{1, 4, 5, 6, 17, 18, 27, 33, 36, 37, 41, 42, 54, 55}	{5a}
121	{1, 5, 17, 18, 21, 27, 33, 34, 35, 36, 37, 54, 55, 56, 57}	{11b}

122	{1, 4, 5, 17, 18, 25, 31, 33, 34, 36, 37, 38, 39, 54, 55}	{14f}
123	{1, 3, 4, 5, 6, 19, 21, 22, 27, 36, 37, 54, 55, 56, 57}	{1a_2}
124	{1, 4, 5, 6, 17, 18, 24, 26, 35, 36, 37, 44, 45, 54, 55, 56, 57}	{8b}
125	{1, 4, 5, 6, 17, 18, 25, 26, 33, 34, 36, 37, 38, 39, 54, 55, 63, 64}	{15c}

12.APPENDIX E

Figure 10 - Visualization of the Base of the Competence Space (higher resolution)


{4; 5; 6; 36; 37; 17; 18; 27; 31}
{4; 5; 36; 37; 17; 18; 27}
{4; 5; 6; 36; 37; 17; 18; 23}
{33; 4; 5; 6; 36; 37; 41; 42; 17; 18; 27}
{17; 18; 23}
{41; 39; 38}
{2; 4; 5}
{2; 4; 14}
{2; 66; 4; 5}
{65; 2; 35; 4; 5; 18; 23}
{2; 4; 5; 6; 72; 45; 58}
{32; 4; 5; 72; 29}
{72}
{4; 5; 36; 37; 17; 18; 26; 31}
{1; 33; 34; 5; 38; 39; 17; 18; 19; 54; 55; 27}
{17; 18; 27}
{17; 18; 22}
{32; 4; 5; 17; 18; 21; 22}
{4; 5; 36; 37; 17; 18; 26}
{1; 18; 39; 17}
{1; 33; 34; 35; 5; 36; 37; 17; 18; 21; 54; 55; 56; 57; 27}
{1; 4; 8; 72; 16}
{1; 33; 5; 38; 39; 41; 42; 17}
{1; 33; 4; 5; 6; 36; 37; 41; 42; 17; 18; 54; 55; 27}
{1; 17; 18; 55; 54; 23}
{1; 4; 5; 38; 7; 39; 17; 18; 54; 55; 27}
{17; 18; 21}
{17; 18; 26}
{54}
{1; 4; 5; 17; 18; 54; 55; 27; 31}
{1; 5; 54; 55; 56; 25; 57}
{1; 34; 33; 36; 5; 37; 54; 55; 56; 57; 27}
{1; 11; 12}
{1; 5; 23; 54; 55; 56; 57}
{1; 11; 13}
{1; 11}
{1; 4; 5; 17; 18; 54; 55; 56; 57; 26}
{56; 54}
{1; 36}
{1; 36; 5; 37; 10; 17; 18; 21; 54; 55}
{1}
{1; 17; 18; 54; 55; 24}
{4; 5; 6; 36; 37; 17; 18; 26}
{1; 5; 54; 55; 24; 57; 56}
{1; 10}
{1; 36; 5; 6; 37; 17; 18; 55; 54; 25}
{1; 33; 34; 4; 5; 36; 37; 17; 18; 21; 54; 55; 24}
{1; 36; 5; 37; 17; 18; 54; 55; 26}
{64; 1; 5; 6; 17; 18}
{1; 17; 18; 54; 55; 25}
{60}
{1; 18; 17; 25; 4; 5; 6; 36; 37; 56; 44; 45; 17; 18; 54; 55; 24; 57; 26}
{1; 4; 5; 36; 7; 37; 10; 54; 55; 56; 57; 26; 27}
{1; 33; 34; 4; 5; 36; 37; 17; 18; 54; 55; 24; 57; 56; 60; 61}
{1; 36; 5; 37; 10; 12}
{1; 36; 37; 5; 54; 55; 56; 57; 27}
{1; 4; 5; 60; 61}
{64; 1; 33; 34; 4; 5; 6; 36; 37; 38; 39; 17; 18; 54; 55; 25; 26; 63}
{1; 4; 5; 38; 39; 17; 18; 54; 55; 25}
{1; 33; 4; 5; 6; 36; 37; 41; 42; 17}
{1; 35; 36; 5; 37; 18; 54; 55; 56; 57; 26; 27}
{1; 4; 54}
{1; 35; 4; 5; 36; 37; 18; 19; 54; 55; 56; 57; 26; 27}
{1; 4; 5; 38; 7; 39; 17; 18; 54; 55; 25}
{32; 1; 4; 5; 36; 37; 17; 18; 54; 55; 25; 26; 5}
{1; 4; 5; 38; 39; 17; 18; 54; 55; 25; 31}
{1; 4; 5; 36; 37; 8; 41; 42; 15; 17}
{1; 4; 49; 5; 44}
{1; 4; 5; 36; 37; 17; 28}
{1; 4; 6; 8; 16}
{1; 33; 34; 4; 5; 36; 37; 38; 39; 17; 18; 54; 55; 25; 31}
{4; 5; 17; 18; 25}
{1; 33; 4; 5; 36; 7; 37; 41; 42; 17}
{1; 4; 5; 7; 8; 16}
{40}
{1; 3; 4; 5; 36; 37; 28; 29}
{1; 3}
{4; 5; 6; 18; 17; 18; 25}
{1; 3; 4; 5; 6; 36; 37; 19; 21; 22; 54; 55; 57; 56; 27}
{1; 4; 5; 52; 53}
{24; 17; 18; 19}
{1; 36; 37; 5; 52; 53}
{1; 3; 4; 5; 6; 36; 37; 19; 41; 42}
{62}
{1; 3; 33; 34; 5; 6; 36; 37; 41; 42}
{1; 3; 4; 5; 6; 36; 8; 37; 41; 42; 45; 19; 58}
{64; 5; 70; 6; 19}
{1; 4; 5; 6}
{1; 4; 5; 6; 36; 37; 48; 49}
{1; 4; 5; 6; 36; 37; 46; 47}
{1; 4; 5; 50; 51}
{69}
{1; 33; 4; 5; 6; 36; 6; 37; 41; 42; 45; 58}
{1; 4; 5; 6; 17; 28; 29; 30}
{1; 4; 5; 6; 8; 45; 58}
{50}
{17; 18; 20}
{1; 33; 4; 5; 6; 45; 58}
{49}
{48}
{1; 4; 5; 46; 47; 50; 51}
{59}
{1; 4; 5; 6; 7; 9}
{74; 75}
{1; 4; 5; 6; 15}
{74; 76}
{74}
{67}
{1; 4; 5; 6; 7; 9; 16}
{74; 77}
{1; 4; 5; 6; 15; 16}
{43}
{70; 4; 5; 30}
{4; 5; 70; 7}
{19; 4; 5; 70}