The Impact of Knowledge Management to Online Education: A Database Course

JOZEF HVORECKY
Vysoká škola manažmentu / City University of Seattle, Bratislava, Slovakia

Abstract. Designing a Database Management course for Information Systems Management students requires understanding its future graduates' specific position between Management and Information Science. They should become database-literate "interpreters" – individuals effectively acting as "glue" and enhancing customer-developer collaboration during business application development. The managers as the future users have sufficient tacit knowledge about the functions to be executed by the planned application but are unaware of the accuracy of their description requested by programmers to make perfect procedures and user-friendly environments. The programmers have explicit knowledge necessary for programming the applications but lack a picture of their business logic.
Using Knowledge Management principles, the author developed a Database course introducing a balance between explicit knowledge (preferred by programmers) and tacit knowledge (favored by managers). The paper describes its philosophy and articulates the course's features supporting online education.

1 Introduction

Among educators, there is a consensus around the necessity of computer awareness for university graduates. Because databases are at the heart of computerized information systems, database courses must become a part of the literacy of managers. A typical example is the introductory Database Management course for Information Systems Management students. In our understanding, its objective is to educate database-literate "interpreters" – the individuals on the watershed between Management and Computer Science – future professionals effectively collaborating during business application development. We support the following statement: "*Single semester database courses tend to focus on the relational model thus restricting topics forcing the inclusion of the others. Once the relational model is selected, the relational algebra, the E-R model, SQL and normalization are included in the curriculum and a particular database management system is used for the implementation*." [7]. It states the ceiling of any introductory database literacy's course as novices can hardly be expected to learn more.

Despite such limited goals, achieving them is not easy. The experience shows (see e.g. [2]) that the absence of mutual understanding between customers (managers) and programmers (developers) is a frequent cause of software development failures. The two partners differ in their dominant knowledge about the future software products:

- The managers (as the future users of the designed application) have sufficient tacit knowledge about the functions executed by the planned application but are unaware of the expected accuracy of their specifications. Only precise and well-specified design allows the programmers to make suitable procedures and user-friendly environments. The managers are rarely trained to delineate them.
- The programmers have explicit knowledge necessary for the procedure programming but lack a picture of their business logic. As a result, often they simply guess their customers' expectations.

In our opinion, the problems are caused by an inappropriate teaching methodology. Typical database textbooks address programmers' needs i.e. they start with very abstract topics and concentrate on SQL concepts. For example, Connoly & Begg [1] explains their material by starting from conceptual, through logical to

physical database design. It takes two thirds of their book before databases in their real-life meaning are discussed. The book's size (1375 pages) will deter almost any non-computer-science students from the course. Even more practically-oriented books Van der Lars [10] and of Silberschatz, Korth & Sudarshan [8] with their 1 056 and 1376 pages respectively, function similarly. One can hardly expect students who do not plan to become database professionals to consume such volumes of explicit knowledge. On the other hand, textbooks addressing Management students (e.g. [9]) mostly speak about business applications at a very general level. They discuss their potential commercial value and do not explain their inside mechanisms. They usually do not mention that the business logic must be implemented within them to a high level of detail and precision. The students therefore learn very little about what to say to programmers when describing their planned software product.

The author designed and developed a textbook comprising the database concepts and loosening the ties of formal notation but not losing the core knowledge necessary for effective development of database applications. Our educational methodology balances students' tacit and explicit knowledge based on the following assumptions:

- Only a selection of database concepts is presented. It is rich enough to allow comprehension of the computational power of the totality of concepts and of the relationships between them.
- The concepts are organized from the simplest to the most difficult.
- Whenever possible, formal notation is replaced by a semi-automated developer support.
- The problems offered to students come from areas familiar to them.
- Relationships between natural language and database constructs are underlined; the links between informal and formal concepts are explained.
- The limited scope of the knowledge presented is explicitly acknowledged. Students become aware that what they get is only a basis for the acquisition of the more substantial skills necessary for relational-database-oriented thinking and reasoning. They are advised to cooperate with database professionals for the solution of tasks beyond the scope of the course.

Chapters 2 and 3 of this paper introduce Knowledge Management as the backbone of our theory. Chapter 4 gives the reasons for our choice of Microsoft Access for implementation. Examples of problems and learning strategies are in Chapter 5. Chapter 6 outlines tacit and explicit knowledge addressed by the course. The paper ends with Conclusions.

2 Knowledge Management Implications

In accordance with the theory of KM, our knowledge has two main forms. A portion called explicit knowledge can be demonstrated using facts, formulae, instructions, drawings, and similar means. SQL commands and rules of normalization are examples. The rest of our knowledge is informal, stored only in our brains and can be termed "tacit" – often, it is gained unnoticed through our day-to-day practice. The ability to form an SQL command corresponding to a user's question or the ability to recognize violations to normalization rules belong to this category. Both types are interrelated and develop simultaneously as Nonaka and Takeuchi's SECI model [6] indicates – see Table 1.

Figure 1. The SECI model

	TACIT KNOWLEDGE	EXPLICIT KNOWLEDGE
TACIT KNOWLEDGE	**Socialization**	**Externalization**
EXPLICIT KNOWLEDGE	**Internalization**	**Combination**

The most common means of development of tacit knowledge is by face-to-face communication. The process in Table 1 is specified as *Socialization*. Generally, it is performed by interpersonal communication and/or intrapersonal insights. Socialization is the most traditional form of learning and it occurs in every community. Nowadays, it is still much more favored in the humanities than in natural sciences and technology. The latter fields prioritize formal notations such as mathematical or chemical formulae, technical drawings, etc. The reasons for this prioritization lie in the vagueness and ambiguity of verbal expressions.

Transfer of tacit knowledge to explicit knowledge is called *Externalization*. Its outcomes (numbers, texts, graphs, formulae, computer programs, and similar forms) create a basis for the wider distribution of knowledge as the "dialogue" between the author of the particular piece of knowledge and its consumer. The SQL language and Entity-Relationship Diagrams (ERD) are examples of the commonly accepted notations of database concepts. Thus writing queries is an externalization of the programmer's ideas. Usually, the database management system (DBMS) is the consumer of this formal notation. Nevertheless, people have to understand them as well, otherwise they are unable to formulate them properly and modify them when necessary. As every database teacher knows, novice programmers' interpretations of their SQL commands often differ from the DBMS' ones. Such mistaken notations results in partially or completely incorrect executions. Thus, it is preferable to have a tool allowing the expression of the same intention using a less formal system (closer to common sense). Query-by-example (QBE) languages are examples of such simplification for query creation.

The pieces of knowledge expressed in their formal notation can be processed by their receivers. Such *Combination* may lead to new knowledge. In databases, knowledge encoded in formal notations is executed by the DBMS. The combination superposes the user's commands on data stored in the memory. Nevertheless, machine-performed Combination represents only a part of all these activities because people constantly have to use their tacit knowledge to specify what new knowledge is to be extracted and how. The correct result can only be received when the database contains required data and a query corresponding to the user's question.

In the last stage, people try to interpret the outcomes of their activity and want to comprehend them. Through *Internalization*, the new piece of knowledge becomes an integral part of our individual knowledge. The learners start to understand the effects of their activities and make them ready for their future applications in analogous situations.

Figure 2. A query creation from the point of view of SECI model

	TACIT KNOWLEDGE	EXPLICIT KNOWLEDGE
TACIT KNOWLEDGE	*Socialization* **The idea of an intended function**	*Externalization* **Its formal expression using SQL**
EXPLICIT KNOWLEDGE	*Internalization* **The comparison of the idea with the result**	*Combination* **Its execution by an SQL engine**

The SECI model helps us to understand the importance of tacit knowledge and its place in education. The course has to include all four SECI components and to run in all three modes: inside quadrants, clockwise and perpetually. It also says that the main aim of learning is knowledge internalization. For example, excellent programmers are not those capable of writing syntactically correct SQL commands but those able to apply any notation accepted by DBMS and (with over time) more and more effectively in accordance to the desires of their potential users.

Let us consider the SECI processes during a query creation (Figure 2). The aim of education is to build "a diagonal identity" i.e. to make students confident in creating queries that are equivalents of their intentions and, as such, their computer execution gives expected results. Whatever is in the user's mind, it will also become the subject of a query. To form the diagonal identity, the SECI loop must function perfectly. Thus, the teaching methodology must enforce the processes the students struggle with.

The principal problems of non-computer-science student are caused by their low ability to form a correct syntax reflecting the semantics stored in their minds. After the execution, they should compare its results with their presumed computer outcome and find discrepancies between them, to disclose their reason and to fix the bug(s). These activities require precise thinking accompanied by the application of a formal notation, the ability to comprehend and interpret data. None of the skills is popular among students of humanities. Thus, this group of students the most frequently fails in the Externalization and Interpretation stages.

In computer-science student groups other problems appear: Problems solved in traditional (computer-programmer-oriented) introductory database courses rarely touch on business logic and its expression by means of SQL. The students are mostly required to transfer exactly specified problems into their SQL equivalent. Thus, they hardly think about the problem formation and about building its informal (mental) solution in a stepwise manner approaching the ideal business strategy. They are capable of forming syntactically correct queries but do not care much about their meaning (and potential business benefits) for their users. Thus, they are not trained enough to face problems typical for the Socialization stage.

Correct applications of the above implications require experience and extensive know-how. In on-ground classes, these can be transferred from educators to learners by face-to-face communication. The same gap makes teaching of the identical course in online classes more difficult. For that reason, the methodology used for on-site classrooms must be expanded by additional dozes of tacit-knowledge-oriented activities.

3 Choosing a DBMS

Microsoft Access has been chosen as the carrier of the practically oriented part of our course. There are several reasons behind the decision:

- It is a part of Microsoft Office Professional. The majority of students has implemented it long before the course and do not need to make any additional arrangements. Many external students can access it at their workplace, too.
- Using DDL in Access is unnecessary as its Table Design View allows forming tables in an equipollent replacement of it. It offers much more legibility and allows faster development.
- Access also contains tools supporting query design and development, for building entity-relationship diagrams, and for building user-friendly environment. They allow the production of high-quality databases without paying unnecessary attention to formal notation. This allows changing the proportion between explicit and tacit knowledge. The students can concentrate more on understanding the roles of these components and on the interrelationships between them and do not need as much attention to the SQL syntax details as usual in other DBMS's.

All these features enhance the development of students' tacit knowledge because they allow them to concentrate on understanding and optimization of the development of particular elements and the simplification of debugging. On top of that, the computational power of Access is sufficient for many small and medium enterprises – an environment typical of that in which the majority of Management graduates work. Students therefore see the applicability of their knowledge from the very outset.

4 An Overview of the Methodology

The SECI model also suggests a stepwise accumulation of knowledge – the newer pieces have to be digested and superposed on well-internalized older ones. It indicates that introductory courses should concentrate on the understanding of the expressive power of formal notations, their relation to the students' ideas and on their internalization. It should also address students from two distant fields – one with intensive Socialization learning approaches and the other preferring Externalization approaches. For the first group it delivers an unusual amount of explicit knowledge; for the other it may seem "extensively talkative". For that reason, our main aim has been to find a balance between tacit and explicit knowledge and to contrast it.

An authentic method of internalization is "learning by doing". Presumed explicit knowledge can be delivered in different formats presuming that it will lead to appropriate tacit knowledge. Owners of relevant tacit knowledge can easily transfer their working habits and skills to new notations – just by saying to themselves: "*I want to do this and this. How do I express my idea using the new notation?*" We therefore do not see obtaining explicit knowledge as a goal but as a tool. Those who have spent a few decades in the field of ICT will likely agree.

In our course, we concentrate on the following blocks of concepts:

- The composition of well-protected structured data;
- Searching for requested information;
- Building an entity-relationship universe;

- Design and creation of a user-friendly environment.

The SECI model also suggests a circular progression. In the "small" circles, the stages are studied sequentially and "separately". Then, every stage moves seamlessly to the next one to form a large circle.

4.1 Data and Metadata

In the first block, the students first learn that databases are just another – better structured – description of our surrounding world. Data tables are then presented as an appropriate storage for records. The concepts of attributes and their data types are also introduced. Finally, the metadata and their role in prohibiting incorrect data input are discussed. As similar restrictions are difficult to implement in spreadsheets, the students become aware of the superiority of using databases as tools for long-term data storage. The students become familiar with more and more complex input restrictions – starting with specifying relevant data types and ending with restrictions binding several attributes. All tasks are posed in natural language in order to develop connections between student's tacit and explicit knowledge. For example, in a gallery database we start with specifying the data types for authors of paintings, their titles, price, years of origin and presence/absence of insurance. In the second round, one-attribute restrictions are trained as "*The painting title is an obligatory attribute value and must not exceed 35 characters"*. Finally, multiple-attribute restrictions are introduced as "*All paintings priced over 1500 € must be insured*". To make the learning loops perpetual, the students are informed that professional programmers are capable of forming even more complex restrictions – and advised to consult their future input restrictions with them.

The concept of the primary and secondary key is also introduced in this round but its role is purposely reduced to being a reference between the table with "fully-fledged" records and a supporting table containing e.g. days of the week. The explanation of their true role in entity-relationship diagrams is postponed until multiple-table databases are discussed.

4.2 Queries

In the second block, the data in one-table databases are searched for requested information. The tables cover common real-life problems such as personal records, book libraries, stock-keeping, pizza delivery, car rental and others. The tables are not always normalized. To avoid confusions, only questions not leading to conflicts caused by anomalies are posed. Queries address typical managerial inquiries such as the aggregation of data and their grouping. The number of keywords has been reduced to an optimal minimum allowing selection of records with specific properties. Again, all tasks are formulated in a natural language. The students are trained to solve their tasks using a stepwise approach. First, they are asked to prepare a query corresponding to the question: "*How many cars have not been returned in time?*" Then, the students are asked: "*Identify them*". The second response requires a nested query, i.e. the problem is substantially more difficult. As they know their number, they can verify the correctness of their outcome more easily. Another feature helping our students to verify their results is small table size. The tables mostly do not contain more than 20 records. Again, we warn our students that their list of keywords is far from complete and the table size does not correspond to that of real-life databases i.e. there is a lot of room for expanding their knowledge.

4.3 Normalization

In the third big block, the reasons for splitting non-normalized tables into several simpler ones are shown. This necessity is demonstrated in a very practical way. Examples of databases with anomalies are shown and questions which cannot be properly answered are asked. All queries proposed by students generate incorrect outputs. Then we search for a transformation that could produce correct ones. We "rediscover" the role of primary and secondary keys and their potential to build relationships between tables (entities). We also study their role in the entity-to-entity cardinality. We pay special attention to forming entity-relationship diagrams and in "reading" (i.e. in interpreting) them. By using practical examples we show how ERDs can be used for understanding producer-consumer chains, family relationships etc.

The third block ends with queries over several tables (JOIN and UNION) and with action queries. Then we point the students' attention to the fact that all query operations can be successfully applied to the joint data. We stress the fact the JOIN and UNION operators form "virtual one-table bodies" so all their knowledge on query design can be directly applied. Another loop is added because the students start understanding that there is no substantial difference between using query design strategies in one-table and multi-table databases.

4.4 User-friendly communication

The last block shows creation of fair and user-friendly communication. Here, we often refer back. For example, the forms built upon join queries automatically distribute input values into their respective tables without the need for checking the identity of the values of primary and secondary keys. Students start to comprehend higher dimensions of normalization. The feeling "it is a puzzle enjoyed by theoreticians only" is dispelled. They see that Access speeds up the creation of the user friendly environment using the structure of properly-related groups of normalized tables. We return students to their period of innocence in which databases were just black boxes with buttons. We point to the simplicity of their operation and invite them to replicate it in their final projects. As a result, our students design and develop simple but fully-functioning database applications with 4-7 tables. To motivate online students, we encourage them to build databases for their day-to-day professional needs.

4.5 Enhancing tacit knowledge

There are two basic methods of enhancing tacit knowledge:

- Discussions
- Learning by doing

Seemingly, discussions address the Socialization stage only. In fact, they apply to all quadrants as the class debates can touch any aspect of the subject:

- Considering alternative ways of metadata descriptions, of query expressions, of table normalization, of the user communication design and comparing their advantages and advantages – all are examples of informal, person-to-person tacit knowledge during Externalization.
- The trial-and-error runs of queries (or user environments) may serve as bases for discussion on the relationship between our intentions and the genuine execution of their notation. They help to understand the processes performed during the Combination stage.

- The Internalization stage is also enhanced. For example, the formulation of queries in natural language supports the control of their correctness. If there is a discrepancy, the students can more easily disclose it because they have a depiction of what it should be.

Each of these processes may run as entire-class discussions, the student's introspections as well as classmates' informal talks during breaks. In all cases, they fulfill their educational function.

In-class activities are based on practicing the course content. The lecturing is reduced and runs in computer labs. In this way, the students can instantly test their proper understanding of their teacher's lecture and ask for further explanation when necessary. Their errors are occasionally presented to the rest of class in order to avoid their repetitions.

Another apt method of enhancing tacit knowledge (widely exploited in virtual classrooms) is our discussion forum. It also facilitates Externalization (as its participants must formulate their ideas legibly), Combination (as every person must compare different views to the subject discussed) as well as Internalization (by valuing these different versions, by evaluating them and then by modifying their own opinions accordingly). For example, data distribution in normalized databases is often anti-intuitive. The discussions allow students to see the ERDs of their classmates, to understand why they can (or cannot) be considered normalized, to discuss the reasons and to see the process of their step by step normalization. We also discuss ways of making the database interior – tables, queries, ERD structures – invisible.

5 Examples of the methodology implementation

As the course addresses non-computer-science students, it concentrates on the topics in which applicability is evident and neglects those whose value is primarily academic. For example, the theory of relational algebra is skipped: only its practical consequences are demonstrated and explained. The entire course is accompanied by four types of support: a textbook [3], a handbook referenced here as Problem Solver [4], thirty lectures on YouTube, and a USB key with databases and PowerPoint slides. Their various roles are specified below. For the above reasons, the book and problem solver use a less formal language than common database textbooks do. Naturally, the informality of the language does not equate with inaccuracy in its constructs or incorrectness of explanation. The register only means that in explanations we relate on the basis of common sense rather than academic purity. For example, when we speak about data organization having a structured form we say: *Records are stored in tables: every record occupies a row – the values of their identical attributes are stored in columns.* We see our figurative description as more student-friendly than the specification of records as tuples which "*are functions from a finite set U of attributes (of the relation) to a domain of values (assumed distinct from U)*" [11].

In Access, tables can be defined using Table Design View. The students can switch between designed tables' Data View and Design View. To relate attributes and their values to real-life objects (e.g. airline tickets) is much easier. This approach completely diminishes the application of the DDL part of SQL. Their metadata are represented as a list of features: "*the price of any airline ticket cannot be negative*", "*the date of the flight must refer to the future*" or "*the name of the departure airport and the name of the arrival airport cannot be the same*" which must be entered into their definition form. Instead of writing complex syntax, the process is reduced to filling in the cells related to particular properties; many of them with pull-down menus. Compared to the approach based on writing equivalent DDL commands, the risk of making unintended errors is reduced. The

students are less stressed and can concentrate more on their problem solving. Our textbook never mentions the existence of DDL as an equally powerful tool is at the student's disposal.

Figure 3: The title slide of the presentation on primary keys

To facilitate tacit knowledge and to point to key principles of table design, a particular part of the textbook contains the QR codes of seven lectures available on YouTube. The PowerPoint slides for all lectures are stored on a USB memory stick available with the Problem Solver. As Figure 3 shows, every lecture can also be accessed via a link and/or QR code placed on the title slide.

The query design also starts without SQL. The built-in tools – Access' Query Wizard and Query Design View are used for first queries. SQL comes into use only when query design previous becomes too complex and/or cannot be completed efficiently. Even then, the students are allowed to combine them using their preferred methods. In [5] an educational experiment comparing the class with a traditional SQL-based course and the one using our new methodology is described. The students of the tested group solved their problems faster and felt more comfortable – whilst the correctness of their solutions was not affected.

Table 1 The table used for the actualization of the *Contracts* table

ActualDataSource	
Car_ID	When
BA 321 PT	0
KE 362 KE	-20
PP 650 DA	-1
TN 409 BG	-10
TT 081 AC	-4
ZA 994 DR	-2

To speed up query creation, every assignment is accompanied by an empty database on the USB. The adjective "empty" means that the database contains all tables and necessary data in them but no queries. The data are "time-sensitive", i.e. they automatically adjust their time values in relation to "today". The actualization exploits the built-in *Date()* function. Its value is the current date. In every database that must be adjusted, there

are one or several tables (usually called *ActualDataSource* or similar - see Table 2). In the table, the *Car_ID* attribute has its standard meaning. The value of the *When* attribute are all non-positive. They will be added to the current date moving all dates of rental i.e. they will be presented to the students within the interval "20 days back from today". The modification is executed by the following update query which is automatically executed after the opening of the database:

```
UPDATE Contracts INNER JOIN ActualDataSource
                 ON Contracts.CarID = ActualDataSource.CarID
SET When = When + Date();
```

The Problem Solver contains 35 pages of problems related to query development and testing. An additional 13 pages of query-related tasks come after introducing multiple-table databases. Queries are discussed in twelve YouTube lectures.

The creation of the global databases structure is exemplified by many positive and negative examples – i.e. successful designs and failures. The ERD-reading skills are stressed because the normalization is not a straightforward process. It must reflect the intended business logic which always affects the distribution of attributes among entities. Figure 4 shows the difference between car-rental agreements in a company with fixed fees and another one with fees which are subjects of negotiation. In the first case, the attribute *RentalFee* belongs to the *Cars* table; in the other one, it is in the *Agreements*.

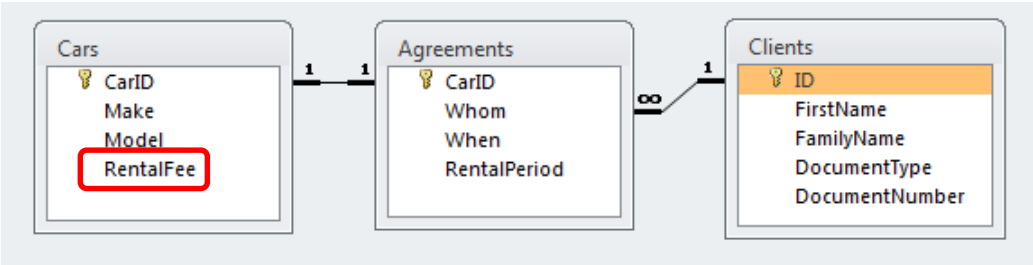

Figure 4a: Rental fee is fixed

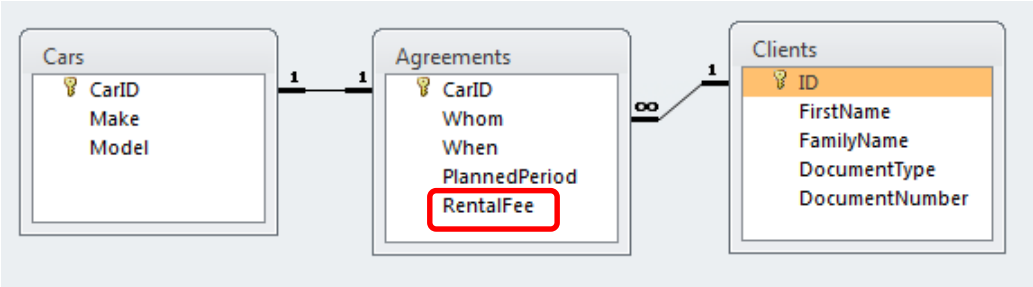

Figure 4b: Rental fee is negotiable

In this way, future managers' are trained to point their attention to the details which may have critical (and sometimes fatal) consequences for the success of their information system. The course is intended to educate qualified users, not professional programmers. Nevertheless, they should understand why the programmers will ask them many silly questions during the development process. This is why our textbook concentrates on similar nuances and exemplifies their influence on future database applications. Assignments in this part of the Problem Solver ask for the design of real-life databases with several tables and their assembly into ERD diagrams. To maintain connection with real life, the applications start in a nursery and end at a cemetery.

Often, the necessity of understanding difference between intuitive representations of databases in spreadsheet tools (like Excel) and in a DBMS is underlined. To emphasize them, we exemplify negative experience with low data reliability:

- Entered without controlling input restrictions by metadata;
- Distributed in tables which are not referenced by primary and secondary keys.

The problem is often discussed in virtual classrooms. In particular, students working with databases implemented in Excel have difficulties understanding the difference.

Finally, the textbook section on user-friendly interfaces discusses not only their importance of their smooth exploitation but also demonstrates differences between well- and badly-designed ones. The Problem Solver contains problems that are accompanied by "empty" databases on the USB key. They now also include queries submitting data which are to be accessed/presented through forms and reports.

6 Conclusions

The course design and its implementation using the above set of tools ware successfully tested. It is used in both on-site and online courses. Naturally, the teacher's practices differ. In the classrooms there is an opportunity of instant reaction to every student's question. Still the students often solve their problems with their neighbors. Specific solutions (both exceptionally good or having typical errors) are presented to the entire class and discussed. The discussion's purpose is to spread these particular pieces of tacit knowledge to everyone. In the case of errors, the students propose their fixation and are requested to explain why it will function so.

On the other hand, the online students start working in isolation. That's why they welcome the less-formal text formulations in the textbook and other materials as well as the opportunity to work with the DBMS which lowers requirements to the usage of formal notation. They can concentrate on practicing problem solving and, consequently concentrate on the development of their skills (i.e. their tacit knowledge). The discussion forum serves as a platform for presenting their ideas, discussing their quality and clarifying the problems. Their examination results and final projects demonstrate that our online course's outcomes are equal to those achieved in on-site classes.

Currently, a similar course for the University of Pavol Jozef Šafárik in Košice is under development. The author sees it as another achievement of VŠM teaching methodology. For the first time in the history of Slovak higher education, a public university applied know-how of a private one. It is a positive signal and demonstrates a growing reputation of VŠM in the field of online education.

Literature

1. Connolly, T. M., Begg, C. E. (2009): *Database Systems: A Practical Approach to Design, Implementation, and Management*, 5th edition, Pearson Education, London, 1375 pp.
2. Ewusi-Mensah, K. (2003). *Software development failures: anatomy of abandoned projects*. The MIT press, 290 pp.
3. Hvorecký, J. (2013a): *Databázové technológie*. Equilibria, Košice, 316 pp.
4. Hvorecký, J. (2013b): *Databázové technológie – podporný učebný text*. Equilibria, Košice, 106 pp.

5. Hvorecký, J., Drlík, M., and Munk, M. (2010): *Enhancing Database Querying Skills by Choosing a More Appropriate Interface.* Education Engineering (EDUCON), Madrid, pp. 1897 - 1905

6. Nonaka, I., Takeuchi, H. (1995): *The Knowledge-Creating Company – How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, London, 284 pp.

7. Robbert, M. A. et al (2000): *The Database course: What Must Be Taught*. SIGCSE Bulletin Proceedings of 31st SIGCSE Technical Symposium on Computer Science Education, March, pp. 403-404. http://ocw.kfupm.edu.sa/ocw_courses/phase3/ICS324-Offering-072/Study%20Materials/What-Must-be%20Taught.pdf

8. Silberschatz, A., Korth, H., Sudarshan, S. (2010): *Database System Concepts*, McGraw-Hill Science/Engineering/ Math, New York, 1376 pp.

9. Turban, E., Leidner, D., McLean, E., & Wetherbe, J. (2008). *Information technology for Management*, (with CD). Wiley.com

10. Van der Lans (2006): *Introduction to SQL: Mastering the Relational Database Language*, 4th Edition, Addison Wesley Professional, New York, 1056 pp.

11. Wikipedia (2013): *Relational Algebra*. http://en.wikipedia.org/wiki/Relational_algebra (Accessed on 23 May 2013)

Contact data:

Jozef Hvorecký, Prof. RNDr. PhD.

Vysoká škola manažmentu, Panónska cesta 17, 851 04 Bratislava, Slovakia
jhvorecky@vsm.sk