Received March 21, 2019, accepted April 12, 2019, date of current version May 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2912522

Reverse SQL Question Generation Algorithm in the DBLearn Adaptive E-Learning System

KANOKWAN ATCHARIYACHANVANICH, SRINUAL NALINTIPPAYAWONG, AND THANAKRIT JULAVANICH
Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
Corresponding author: Kanokwan Atchariyachanvanich (kanokwan@it.kmitl.ac.th)

ABSTRACT Using a traditional e-learning system, when teaching structured query language (SQL) queries in classical classrooms help instructors, to improve the students' SQL skills and learning effectiveness. However several problems in using e-learning as a teaching and learning assistant remain – such as difficulties in differences in learning ability and knowledge level. We solved these problems by applying an adaptation module to our e-learning system. However, we still found it required considerable effort to create enough exercises to make the adaptation effective enough. So, we developed a novel automatic question generating algorithm, named Reverse SQL Question Generation Algorithm (RSQLG), to automatically generate exercises (including both answer and question) from a source database. RSQLG reverses the traditional manual process used previously by instructors. Instead of creating questions and answers for them, RSQLG creates the answers first. The generated exercises are presented to students by applying question adaptation methodology based on student knowledge level in each supported learning objective. We evaluated the learning effectiveness of our approach by using outcome-based learning. After post-test to pre-test scores were compared, we found students using our system improved their scores by 26%. Consequently, the adaptive e-learning framework using RSQLG could be applied in any adaptive or traditional e-learning for a database course to benefit the instructors leading to less effort in exercise management and to improve the learning outcome from the students allowing as much practice as they need.

INDEX TERMS E-learning, adaptive system, automated question generating algorithm, computer-aided instruction, SQL learning.

I. INTRODUCTION

Basic SQL statements (SELECT, FROM and WHERE clauses) are a key foundation when studying practical database courses. Many instructors use an e-learning system to improve the quality of teaching and learning. Applying e-learning benefits both instructors and students, leading to flexibility to review learning materials at any time, ability to access exercises on their own initiative and convenience to check the correctness of student answers. However, teaching and learning through e-learning still faces multiple challenges. Students will receive the same, fixed set of contents and exercises created by instructors, whereas, in reality, each student has a different knowledge level and learning ability. These exercises may not align with or do not cover the learning development of individual learning. As a result, its effectiveness of expected learning outcome was not achieved. In the same class, there might be some students who already know some SQL commands and who might want to learn new commands. There are also students who do not know any SQL commands. If students practice repeatedly with the same set of exercises, their knowledge will not improve because they may answer the questions from memory, without understanding. Also, students may become bored because there is no progression in difficulty and complexity of the exercises. There are also difficulties in tracing which SQL commands the students did not understand. In teaching and learning management, instructors must know how their students should learn in both theoretical and practical dimensions. Therefore, instructors should develop content to suit students individually [1]. For this reason, we developed idea concept of adaptive e-learning which can adapt content, learning sequence and exercises to suit students individually [1]. Adaptive e-learning can help students gain a better

The associate editor coordinating the review of this manuscript and approving it for publication was Anand Paul.

learning outcome than traditional e-learning, which allows only learning in a classroom without assisting systems [2].

However, there are still few studies developing adaptive e-learning systems for fundamental SQL courses. From the researchers' perspective, the implementation of adaptive e-learning in practical courses leads to variety and a large numbers of available exercises to make the adaptation process effective. This is a key problem for instructors due to the effort needed to provide enough exercises to satisfy student needs. Generally, the instructor sets learning objectives for testing each SQL command and writes SQL questions based on these learning objectives. Consequently, the process of creating exercises is time consuming. To solve all these problems, we divided the problem into three tasks. Firstly, we present an algorithm to automatically generate basic SQL questions to be used in an adaptive e-learning system. The algorithm works in reverse, it generates the SQL queries from the answers. Secondly, we present an approach for developing an adaptive e-learning system for SQL courses using the Reverse Generation SQL Question (RSQLG) algorithm. This algorithm generates large numbers of questions with much less instructor effort. Students also see questions which suit their knowledge level and needs. We believe that students will improve their basic SQL skills by using our system. Lastly, we will show that learning outcomes from our DBLearn Adaptive e-learning system exceed those of traditional e-learning systems and our system can adapt to student knowledge level effectively. Our research will also enable instructors to achieve outcome-based learning by teaching a practical database course.

II. RELEATED WORK

A. ADAPTIVE E-LEARNING

E-learning systems are playing an important role in education. Many instructors use e-learning in their courses, including basic database courses. For example, Cvetanovic *et al.* developed an e-learning system for teaching a basic database course in both theoretical and practical aspects [3]. Students can practice their skills with exercises. The system supports almost every learning objective listed in the course syllabus. In traditional e-learning, instructors must create content by themselves. After that, students must practice using the same set of materials. So, some researchers developed a concept of adaptive e-learning which adapts contents, learning sequences and exercises in accordance with the differences in student knowledge and skills [1], [4].

E-learning adaptation techniques try to make traditional e-learning adapt and respond to student differences in knowledge and skill. There are many proposed techniques. Following Brown et al., approaches are separated into two groups [5]. First, adaptive presentation systems, which adapt the method of presenting content to suit student style of perceiving or learning categorized by learning style theory or trained user model [6]–[9]. Second, adaptive navigation systems adapt the learning sequence and suggest the next learning objective that the student should take matching their goals and knowledge level [10]. This technique can be applied using many methodologies. Zemirline *et al.* presented five adaptive navigation techniques – direct guidance, adaptive ordering, link hiding, link annotation and link generation – which can use student actions in the system, such as answer correctness and time used to read content, to make suggestions [11]. Computer adaptive testing (CAT) adapts the number of questions, their difficulty and learning objective of exercises to suit student ability [12], [13].

There are few studies on adaptive e-learning for database courses. For example, Pahl and Kenny. presented an approach to correcting, providing feedback and presenting recommended questions for programming courses [14]. This approach uses language grammars and syntax trees to process the feedback. Mitrovic presented SQLT-Web, which used constraint-based modeling to form knowledge models from submitted answers [15]. When a student submits answers, the system analyzes answers and mistakes by comparing a student's answer with the correct answer using defined constraints. Then it sends an appropriate action such as repeat the same question, log out and go to next question. Nalintippayawong *et al.* proposed an approach to predict student knowledge using mistakes from answers [16]. However, all researchers [14]–[16] still do not prepare enough questions to lead to knowledge improvements.

B. ALGORITHM FOR AUTOMATED QUESTION GENERATION

Using an algorithm to automatically generate questions is a new approach to lighten the instructor's workload and enhance sustainable e-learning practices for students. We believe that this is the first work that proposes algorithm generated SQL questions and answers. Reviewing generation algorithms in related areas, we found four approaches. Firstly, "questions by mutation" mutates the input into a new object or answer. For example, for English language courses, Lee *et al.* used sets of errors found by analysis of existing corpora and generated questions containing common errors which students were required to identify [17]. Similarly, Funabaki *et al.* created debugging questions for programming courses [18], [19]: their algorithm creates bugs in JAVA source code using one (or more) of three methods – command deletion, variable swapping and command insertion. Secondly, "questions by keyword" extracts keywords from the input and generates a new question from the extracted keywords. In mathematics courses, Nandhini and Balasundaram generated mathematical problems for students with learning difficulties: to enable students to understand various problems, the keywords in the original problem were used to generate new several problems that require solutions which differ from that of original problems [20]. Liu *et al.* helped students improve their literature reviews by parsing sentences from their reviews, extracting key phrases and finding appropriate sentences from Wikipedia pages, then forming questions about those sentences [21]. These questions help students improve their reviews without needing instructor responses. In, "questions from input", Jain *et al.* generated questions from the input image [22]. Their research can be applied in many fields, e.g. educational study, driving assistance and chatbots. Lastly, Abdul Khalek and Khurshid's "questions from metadata" generates new SQL queries from the database schema for testing database performance [23]. However, their work did not consider data in tables or present any way to generate a text explanation of the query being used as an SQL question in exercises.

Previous research basically converts a source element into a new element (question). The source element can be either the answer, such as a grammatically correct sentence and bug-free source code, or something more complex, such as an input image or student's literature review. We applied this concept in our study to generate SQL questions and answers. The data in a database is our source element. Since the exercises in a basic SQL course query data from the database, we will extract data from the database to generate, as output, the SQL query. Then, we augment the output query with a text explanation of the query.

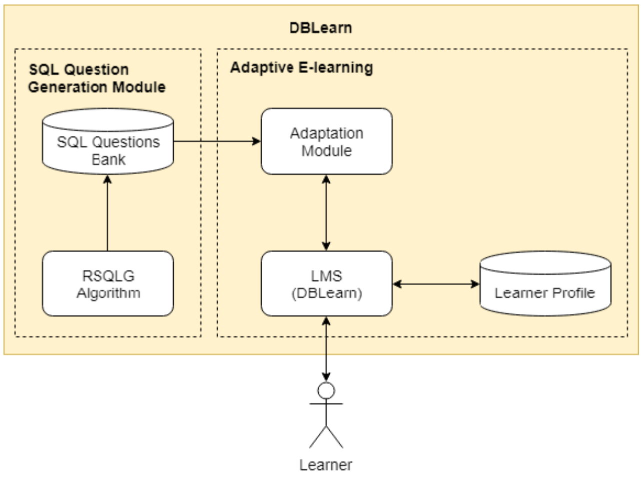


FIGURE 1. Structure of adaptive e-learning (DBLearn).

III. OUR ALGORITHM

The RSQLG algorithm was adopted in the adaptive e - learning shown in an architecture of adaptive e-learning (Fig. 1). Our system has two main parts – SQL question generation and adaptive e-learning system. The SQL question generation module generates SQL exercises with questions and answers and stores them in the questions bank. The exercise generation step is needed once per course when initiating the system. The generated SQL questions are stored in the SQL questions bank. An adaptive module uses a question from the bank and the student's profile from the learning management system (LMS) – DBLearn system [16] - to process and deliver a set of suitable SQL exercises via the LMS interface. After the student has finished the exercises, the system analyzes the results and predicts whether the student understood each SQL command or not. The system will store the predicted result to the student's profile ready for his or her next use of the system.

A. SQL QUESTION GENERATION ALGORITHM

The generation of SQL questions process uses our SQL question generation algorithm (RSQLG). This algorithm operates on the input database and converts it into SQL exercises. The exercise has both questions and answers. Each SQL answer is a query formulated from database metadata. Then, each SQL answer is translated into a text explanation and used as the SQL question for students.

Thus, we applied two approaches i.e. "questions from keywords" (SQL question generating process) and "questions from metadata" (SQL answer generating process) to our algorithm. The two approaches are discussed further in section V.

B. ADAPTIVE E-LEARNING

1) LEARNING MANAGEMENT SYSTEM (LMS)

The study used DBLearn system [16] as LMS. DBLearn is an e-learning tool designed specifically for database courses. The system supports student practice of SQL queries and is able to check the correctness of the student answers. It also automatically gives error feedback to students. DBLearn is the interface between students and the adaptation module; it sends data retrieved from the student profile to the adaptation module. The adaptation module processes it and stores it in the student profile. The student profile stores the student history, e.g. exercise results, learning progression and knowledge level for each learning objective.

2) ADAPTIVE MODULE

Since we aimed to adapt methods currently used in e-learning SQL courses, so that students could practise online until they are skilled, we considered that the content adaptation technique might not be optimal for practical courses. Thus, we should adapt the exercises the students practised, to make sure they understood the SQL commands and achieved the learning objectives of the course. For this practical course, appropriate adaption strategies are learning sequence adaptation, knowledge suggestion and adaptive testing.

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

FIGURE 2. Example of SQL query using LIKE command.

The exercises or tests adaptation [12], [13] is the algorithm to evaluate and predict student knowledge level in each learning objective using the results from previous recorded answers. The exercises with appropriate difficulty degrees, i.e. "not too easy"; "not too difficult", etc., will be presented to students. As the student attempts more exercises, the algorithm can predict their knowledge level more accurately from the answers. For the SQL course, the difficulty level is based on the complexity of the query. For example, a correct response for the LIKE command exercise is shown in Fig. 2.

```
SELECT column_name(s)
FROM   table_name
WHERE
       column_name1 LIKE pattern
       AND
       Column_name2 > value
ORDER BY column_name;
```

FIGURE 3. Example of SQL query using LIKE with AND commands.

The query in Fig 2 is basically very simple and easy, but other clauses, e.g. AND, OR, can be added making it more difficult – see Fig 3, so we classified exercises by their associated learning objectives. The algorithm can suggest more difficult and complex exercises which use multiple commands, as needed.

IV. DBLearn – RSQLG CONCEPT AND FRAMEWORK

A. DBLearn – RSQLG CONCEPT

The traditional process of generating SQL exercises is shown in Fig. 4. The instructor usually begins with exploring the database schema and the data itself. Then, a question is posed based on a query that a student should understand and aligned with a learning objective. Next, the instructor specifies the correct SQL query to answer the question. Finally, this query is validated using the real database to determine if the result is returned as specified in the question or not.

Our RSQLG concept generates SQL exercises by reversing the process in the traditional method. We choose a set of data and, from that data, infer the SQL query that will produce it: this is the 'reverse' step in our algorithm. Finally a text explanation of the SQL query is output – see Fig. 5.

B. DBLearn – RSQLG FRAMEWORK

Our RSQLG framework has two parts. Firstly: Database pre-processing extracts database features necessary for generating questions. Secondly: the RSQLG algorithm has three main steps – a) query metadata generation, b) output of the SQL and c) text explanation generation. The input for the RSQLG algorithm is the pre-processed database with metadata and question settings. These are SQL commands and required question styles or patterns. RSQLG returns a complete SQL exercise with question and answer as well as the input database. This algorithm currently supports basic SELECT statements which allow data selection from one table and a WHERE clause including the relational operators: BETWEEN, IN, IS NULL, LIKE and logical operators. The process for RSQLG is shown in Fig. 6.

All possible exercises from the input database will be generated before students log in to the system to prevent slow response time problem while doing the exercises. We decided not to generate SQL questions one by one after students chose the learning objective because generating advanced SQL exercises is slow if the query contains more than one condition. These conditions cannot use pre-processed data from database metadata due to differences in details. Then, we need to confirm that the generated query is valid by testing it on the actual database by using the MySQL web service [24]. Finally, all possible exercises are ready to use by the DBLearn system.

1) DATABASE PRE-PROCESSING

RSQLG pre-processing extracts the data and metadata necessary for SQL query creation, e.g. database schema, table schema, attribute data type, attribute data length, attribute constraints, relationship and key. This reduces the time to generate exercises, because RSQLG will not need to connect to the real database, whenever it needs some data. Furthermore, we generate some aggregated data, e.g. the maximum and minimum values of numeric attributes when the SQL query contains a WHERE clause with relational operators ($>$, $<$, $=$); common strings, date and time attributes when the query contains a WHERE clauses with a LIKE operator. This data will be input to the RSQLG algorithm.

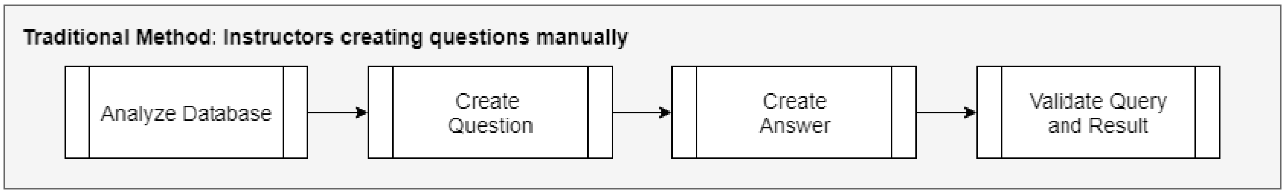

Traditional Method: Instructors creating questions manually
Analyze Database
Create Question
Create Answer
Validate Query and Result

FIGURE 4. Process of manually generating SQL exercises.

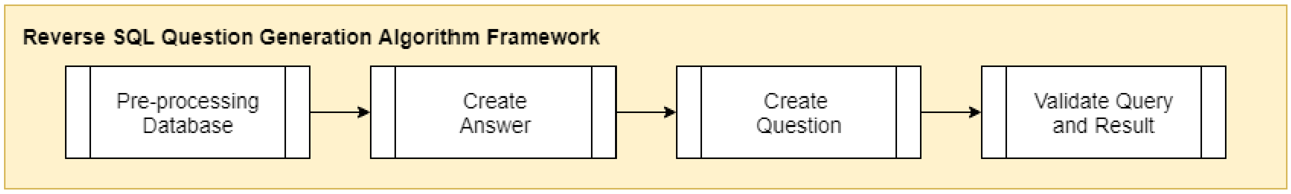

Reverse SQL Question Generation Algorithm Framework
Pre-processing Database
Create Answer
Create Question
Validate Query and Result

FIGURE 5. Concept in generating SQL exercises using RSQLG.

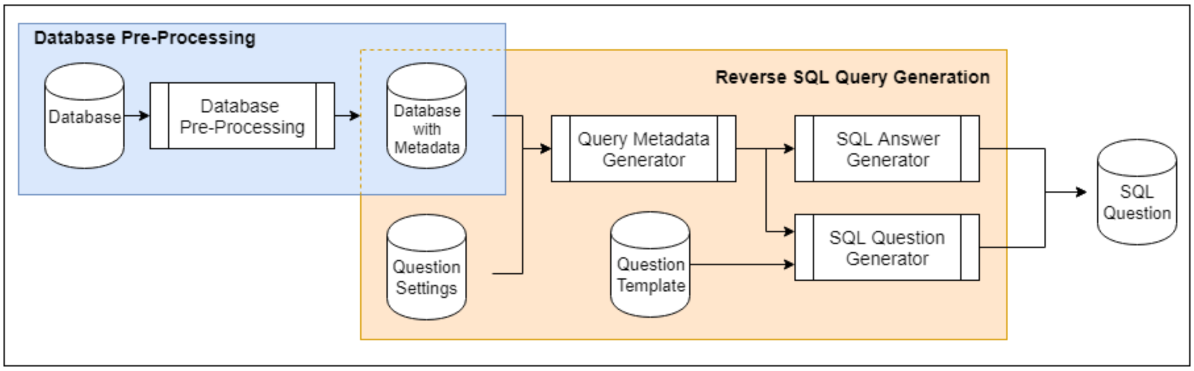


FIGURE 6. Framework of RSQLG algorithm.

```
{
    "command": "SELECT",
    "columns": [
                {"table": "employees", "attr": "name"},
                {"table": "employees", "attr": "salary"}
           ],
    "condition": {
                "exp": [
                      {
                            "table": "employees",
                            "attr": "salary",
                            "op": "<",
                            "value": "25000"
                      }
                  ],
                "lop": null
            },
    "order": [
              {
                "table": "employees",
                "attr": "name",
                "order": "DESC"
              }
           ]
}
```

FIGURE 7. Query metadata example.

2) QUERY METADATA GENERATOR

An example of SQL query metadata - in the JavaScript Object Notation (JSON) - for generating SQL queries and accompanying text explanations is shown in Fig. 7. The SQL queries are generated in sequential order by SELECT statement execution order (FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY, LIMIT) [25]. The Query Metadata Generator algorithm works on basic SQL clauses – FROM, WHERE, SELECT and ORDER BY clauses.

The first key in the JSON object is "command", which selects the type of metadata, in Fig. 7, the metadata is used to create a SELECT command, the "columns" key defines which fields will be presented. The value of this key is an array of JSON objects containing the table and attribute name for each field. The "condition" key describes the WHERE clause of the query: it is a JSON array of expressions (key "exp") and logical operator for joining the expressions (key "lop"). The "exp" array contains JSON objects for each expression. The members of the object are table, attribute, operator (key "op") and value of the expression. Finally, the "order" key defines the ORDER clause. The member is an array of order expressions, each containing a table, attribute and order (ASC or DESC).

Details of the SQL query metadata generation follow.

a: FROM CLAUSE (TABLE SELECTION)

Firstly, the database table will be chosen from a list in the input database and sent to the next step (generation of WHERE clause, fields selection and ORDER BY clause). After RSQLG has generated all possible exercises from one table, it will choose the next table in the list for further query metadata generation.

b: WHERE CLAUSE

After the table has been chosen, the algorithm will generate conditions for the query. The operators in the WHERE clause are described in sections b.1–b.6 or, if question setting specifies no WHERE clause, this step will be skipped.

b.1: RELATIONAL OPERATORS

For a WHERE clause with relational operators, the algorithm selects each operator ($>$, $<$, $=$, $>=$, $<=$, $!=$) in turn, coupled with a pair of randomly chosen fields and values for them from the table. The field data type can be either numeric or text. There are some constraints: if the operator is $>$ or $>=$, the chosen value must not be the highest value (for a numeric data type) or the last value in alphabetical order (for text data type). Similarly for $<$ or $<=$, the other extremes must not be chosen. These constraints prevent an empty result being returned from the query. The number of possible generated conditions, $\boldsymbol{Y}$. is:

$$Y = \sum\nolimits_{i=1}^{n} (V_i O - 4) \qquad (1)$$

when

$\boldsymbol{V_i}$ = Number of possible values in field $\boldsymbol{i}$

$\boldsymbol{O}$ = Number of operators (default is 6 - $>$, $<$, $>=$, $<=$, $=$, $!=$)

$\boldsymbol{n}$ = Number of fields in the table that meet constraints

b.2: BETWEEN OPERATORS

For BETWEEN or NOT BETWEEN operators, fields are iteratively select from the table. Again, the data type of can be either numeric or text. The algorithm randomly chooses two or more values for the chosen field to be used with the operator. There are some constraints here also. The chosen values must not be neighboring values in an ordered set and may not be the first and the last values in the same condition. These constraints are created to avoid students being confused, when they write a legal SQL statement, which accidentally returns a NULL, because the test data is sparse. The number of possible conditions for one table is

$$Y = O\left((n-3) + \sum_{i=1}^{n-3} n\right) \tag{2}$$

The default value of $\boldsymbol{O}$ is 2 (BETWEEN and NOT BETWEEN operators).

b.3: IN OR NOT IN OPERATORS

Each field is selected iteratively and coupled with two or more randomly selected values. We arbitrarily restricted the size of the set of values which is the argument of IN or NOT IN to four values. In this case, we can generate

$$Y = \sum_{i=1}^{n} O\left[\binom{V_i - 2}{3} + \binom{V_i}{4}\right] \tag{3}$$

conditions.

The default value of $\boldsymbol{O}$ is 2 (IN and NOT IN operators).

b.4: IS (NOT) NULL OPERATORS

Each nullable field is selected iteratively. A pair of conditions matches the number of nullable columns in the table.

b.5: LIKE OPERATORS

LIKE or NOT LIKE operators iteratively select a text field by looking up the metadata in the pre-processed database and generate a pattern from pre-processed data. The number of possible conditions is determined by the number of generated patterns.

b.6: LOGICAL OPERATORS (AND, OR)

The AND operator links condition A and condition B: condition A is chosen to select more than one record. Then condition B is created from data in the result set of query A. The fields are checked: they must not be primary keys of the table.

For OR, the steps are essentially the same, but condition B is created from data not in the result set of query A.

c: SELECT CLAUSE (FIELD SELECTION)

The fields listed in a SELECT clause can be chosen in two ways. When some fields are specified in the question setting, the algorithm will randomly select one or more (but not all) fields. Up to three fields may be set. If the question setting does not allow field selection, the algorithm selects all fields (SELECT *).

d: ORDER CLAUSE

The ORDER clause can only be generated for a query whose result set has more than one record: Again, we avoid student confusion when an ORDER clause has no effect. The algorithm randomly selects one or more different fields to use in the ORDER clause. Again, we arbitrarily limited the maximum number of fields to be two and assigns ORDER operator (ASC or DESC) to the selected fields.

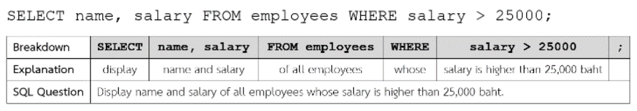
3) SQL ANSWER GENERATOR

After query metadata has been generated, the metadata from Fig. 7 will be processed to generate the SQL query: the basic query statement will have this form:

```
SELECT select_expr
FROM table_references
WHERE where_condition
ORDER BY (col_name | expr | position)
         [ASC | DESC],...]
```

For example, after query metadata from Fig. 7 was processed, the algorithm outputs this SQL query:

```
SELECT name, salary
FROM employees
WHERE salary = 25,000;
```



```
SELECT name, salary FROM employees WHERE salary > 25000;
```

Breakdown	SELECT	name, salary	FROM employees	WHERE	salary > 25000	;
Explanation	display	name and salary	of all employees	whose	salary is higher than 25,000 baht	
SQL Question	Display name and salary of all employees whose salary is higher than 25,000 baht.					

FIGURE 8. Example of output from query extraction and translation.

4) SQL QUESTION GENERATOR

The SQL question generator augments the query metadata output from the generator with a text explanation. This is a natural language description of the query by separating out each part of command and replacing it with natural language. An example is shown in Fig. 8. The table and field names will be retrieved from the database metadata. There is no restriction on the pattern or language of the SQL questions. Instructors can set their own preferences by editing the SQL question templates. In our experiments, Thai language was actually used in the template, but English equivalent examples are shown in Fig. 8. It shows a template based on the query metadata in Fig. 9.

5) ALGORITHM WRAP-UP

Fig. 10 demonstrates a pseudocode of RSQLG algorithm for generating the following SQL exercise. The multiple-line comments describe the step-by-step output in RSQLG algorithm.

Question: Display salary of employees whose first name is Adam and last name is Smith in descending order of their salary.

Answer:

```
SELECT employees.salary
FROM employees
WHERE employees.first_name = 'Adam'
AND employees.last_name = 'Smith'
ORDER BY employees.salary DESC;
```

```
{
  "SELECT": {
    "main": "Display :field of :table :where. :order",
    "flag": {
      "WHERE": {
        "=": ":field is :value",
        "!=": ":field is not :value",
        ">": ":field is higher than :value",
        ...
      },
      "LIKE_PATTERN": {
        "X%": "Begin with :X",
        "%X": "End with :X",
        ...
      },
      "ORDER": {
        "main": "Sort data in :expression.",
        "ASC": "ascending order of :field",
        "DESC": "descending order of :field"
      },
      "all_data": "all data"
    }
  },
  ...
}
```

FIGURE 9. Example of SQL question template.

V. ADAPTIVE E-LEARNING FRAMEWORK

Reading textbooks and participating in a lecture class are not enough for learning SQL in practical courses. It is essential to make students show they have met the objective of the class by carrying out exercises, assignments and quizzes themselves. The selected adaptation techniques are questions adaptation and knowledge suggestion to accomplish the self SQL practice. The adaptive e-learning framework was designed and applied in the DBLearn system shown in Fig. 11.

After the student has logged into the system, the student chooses whether he or she wants to use default configuration suggested by the system or not. If the student has used the system before, the system will retrieve the student profile to suggest the next learning objectives they should take from their historical data and also suggests the number of questions they should practice in a session – default value is 10 questions. For students who have never used the system before, the system will suggest the default questions by degree of difficulty from the easiest to the hardest ones. If they do not want to use the suggested questions, they can choose other learning objectives and the number of questions they want to practice by themselves. Then, the system will retrieve SQL questions from the SQL questions bank, and deliver them to the students. The system will analyze the submitted answers to make a suggestion for the next login. In questions retrieval, the number of questions for each learning objective is calculated using the weight for each learning objective. The weight was defined from the number of questions generated in the SQL questions bank, the learning objective difficulty and the student's knowledge level. The number of questions is calculated by (4).

$$Q_i = \left\lfloor Q_{all} \times \left(\frac{W_i}{W_{all}}\right)\right\rfloor \tag{4}$$

where $\boldsymbol{Q_i}$ = number of questions for learning objective $\boldsymbol{i}$
$\boldsymbol{Q_{all}}$ = total number of questions

```
function generate_exercise(input_database,
                           question_settings):
    //Database Pre-processing
    preprocessed_database = PreprocessedDatabase
                                (input_database)
    preprocessed_database.preprocess()

    //Query Metadata Generator
    query_metadata = QueryMetadata
                          (preprocessed_database,
                              question_settings)
    query_metadata.generate_FROM()
    /*
    * Output:
    * FROM employees
    */
    query_metadata.generate_WHERE()
    /*
    * Output:
    * FROM employees
    * WHERE employees.first_name = 'Adam'
    * and employees.lastname = 'Smith'
    */
    query_metadata.generate_SELECT()
    /*
    * Output:
    * SELECT employees.salary FROM employees
    * WHERE employees.first_name = 'Adam'
    * and employees.lastname = 'Smith'
    */
    query_metadata.generate_ORDERBY()
    /*
    * Output:
    * SELECT employees.salary FROM employees
    * WHERE employees.first_name = 'Adam'
    * and employees.lastname = 'Smith'
    * ORDER BY employees.salary DESC
    */

    //SQL Answer Generator
    sql_answer = SQLAnswer(query_metadata)
    answer = sql_answer.generate_query()

    //SQL Question Generator
    sql_question = SQLQuestion(query_metadata)
    question = sql_question.generate_question()
    /*
    * Output:
    * Display salary of employees
    * whose first name is Adam and last name is Smith
    * in descending order of their salary.
    */

    return ["answer": answer, "question": question]
```

FIGURE 10. Pseudocode of RSQLG.

$\boldsymbol{W_i}$ = chosen weight
$\boldsymbol{W_{all}}$ = Sum of chosen weights

For example, in Table 1, the student chose four learning objectives – SELECT table, WHERE conditions (>, <, =), WHERE conditions (IS NULL) and ORDER BY. The default

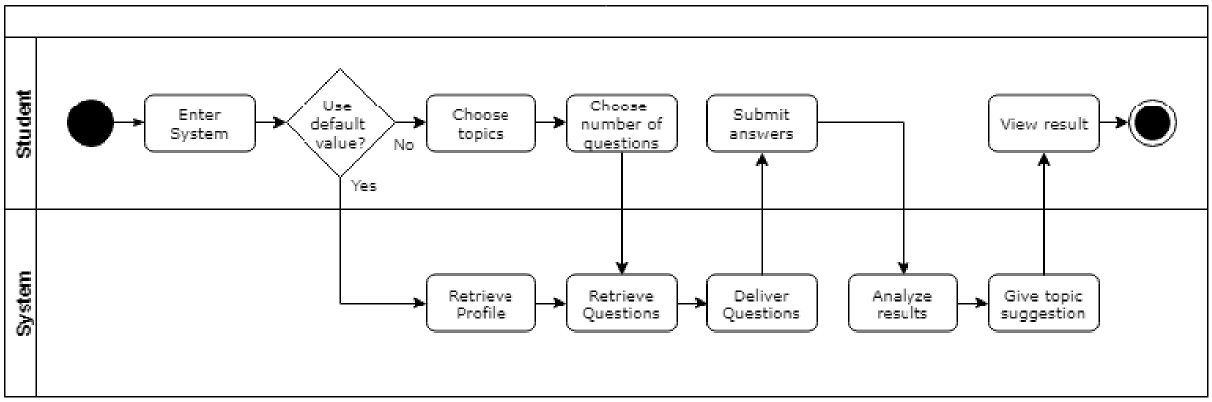
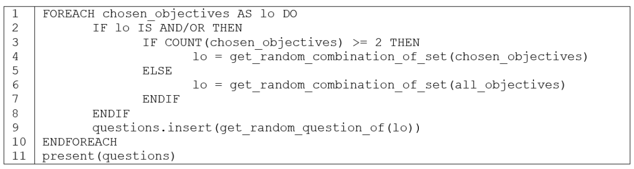


FIGURE 11. Adaptation module framework.

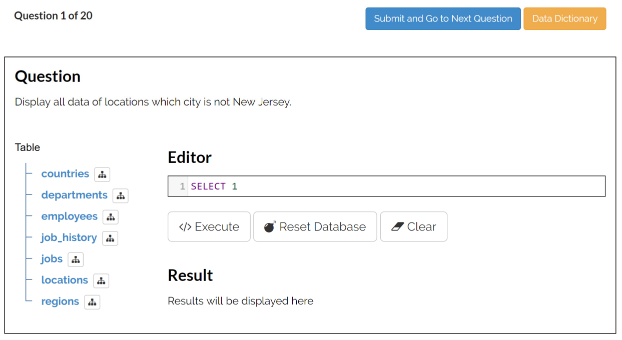
TABLE 1. Example question volume calculation for each learning objective.

Knowledge	Default Weight	Choose?	Chosen Weight	Number of Questions
SELECT table	1	Yes	1	20 * (1 / 9) = 2.22 => 2
SELECT fields	3	No	-	0
WHERE conditions (>, <, =)	3	Yes	3	20 * (3 / 9) = 6.67 => 7
WHERE conditions (IN)	3	No	-	0
WHERE conditions (BETWEEN)	3	No	-	0
WHERE conditions (IS NULL)	2	Yes	2	20 * (2 / 9) = 4.44 => 4
ORDER BY	3	Yes	3	20 * (3 / 9) = 6.67 => 7
WHERE LIKE	3	No	-	0
WHERE AND/OR	3	No	-	0
TOTAL			9	20



```
FOREACH chosen_objectives AS lo DO
    IF lo IS AND/OR THEN
        IF COUNT(chosen_objectives) >= 2 THEN
            lo = get_random_combination_of_set(chosen_objectives)
        ELSE
            lo = get_random_combination_of_set(all_objectives)
        ENDIF
    ENDIF
    questions.insert(get_random_question_of(lo))
ENDFOREACH
present(questions)
```

FIGURE 12. Question retrieval pseudocode.



Question

Display all data of locations which city is not New Jersey.

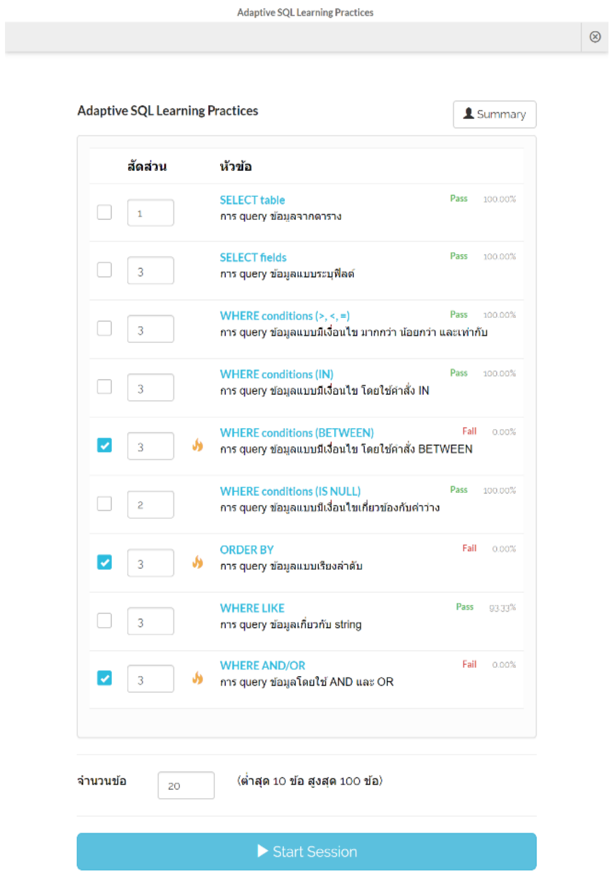
FIGURE 13. Sample screenshot of DBLearn.

weights were 1, 3, 2 and 3 respectively. The system will generate a number of questions based on the chosen weight. In the example in Table 1 the student chose to attempt 20 questions in total and the system calculated numbers of questions for each learning objective using Eq. (4), leading to 2, 7, 4 and 7 respectively.

The system retrieves questions by randomly selecting them from the SQL questions bank. For complicated learning objectives or learning objectives following on from other learning objectives, e.g. ORDER BY and AND or OR operators in WHERE clauses, the algorithm for selecting questions randomly chooses two learning objectives and retrieves questions associated with the basic question having ORDER BY and AND or OR operators in WHERE clauses. The pseudocode is shown in Fig. 12.

After questions retrieval is finished, the system will deliver questions to the student via the DBLearn system. Fig. 13 shows a DBLearn screenshot.

After the student submitted his or her answers, the system will calculate the percentage of questions in each learning objective that the student answered correctly. In the same learning objective, the average percentage of the last three sessions was calculated as a value representing student's knowledge level in each learning objective. Each session of three consecutive sessions must lie in a 7-day interval between this and the previous session. For example, the student did the first session on day 1, the second session on day 5, and the third session on day 17, the system will calculate percentage of correct answers from only the third session. In another example, the student took the first session in day 1, the second on day 10 and the third on day 12. The system will calculate a percentage using the correct answers from only the second and the third session. This constraint was added because it represents the current knowledge



Adaptive SQL Learning Practices

Adaptive SQL Learning Practices

Summary

	สัดส่วน		หัวข้อ		
☐	1		SELECT table การ query ข้อมูลจากตาราง	Pass	100.00%
☐	3		SELECT fields การ query ข้อมูลแบบระบุฟิลด์	Pass	100.00%
☐	3		WHERE conditions (>, <, =) การ query ข้อมูลแบบมีเงื่อนไข มากกว่า น้อยกว่า และเท่ากับ	Pass	100.00%
☐	3		WHERE conditions (IN) การ query ข้อมูลแบบมีเงื่อนไข โดยใช้คำสั่ง IN	Pass	100.00%
☑	3	🔥	WHERE conditions (BETWEEN) การ query ข้อมูลแบบมีเงื่อนไข โดยใช้คำสั่ง BETWEEN	Fall	0.00%
☐	2		WHERE conditions (IS NULL) การ query ข้อมูลแบบมีเงื่อนไขเกี่ยวข้องกับค่าว่าง	Pass	100.00%
☑	3	🔥	ORDER BY การ query ข้อมูลแบบเรียงลำดับ	Fall	0.00%
☐	3		WHERE LIKE การ query ข้อมูลเกี่ยวกับ string	Pass	93.33%
☑	3	🔥	WHERE AND/OR การ query ข้อมูลโดยใช้ AND และ OR	Fall	0.00%

จำนวนข้อ 20 (ต่ำสุด 10 ข้อ สูงสุด 100 ข้อ)

Start Session

FIGURE 14. Sample of DBLearn knowledge suggestion screen.

level better. In that period, the student might have studied from other sources and improved his or her skills. In addition, the students may have not been practicing for long time and forgotten some points. The developed system is shown in Fig. 14.

VI. SYSTEM EVALUATION

A. DBLearn – RSQLG CONCEPT

We measured the achievement of adaptive SQL learning system using the RSQLG algorithm against three objectives:

1) To evaluate the hypothesis: students who use the system had better learning outcomes than students who did not.
2) To assess the accuracy of the knowledge level prediction.
3) To assess user satisfaction.

We tested a population of 152 students who enrolled in a database system concepts course (2nd semester of academic year 2017, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang). There were 42 first and 110 second year students. Since we wanted to assess the learning progress and learning outcome of students in the same background, 110 second years were chosen as samples. In addition, this course is a core course for all second-year students. All sampled students took a basic SELECT command quiz (Quiz #1) which had six questions (25 points total) to assess their basic SQL skill including. This basic SQL command includes SELECT clause, FROM clause, WHERE clause and ORDER clause. The students were randomly divided into two groups based on their quiz #1 score in order that the average scores of both groups were not significantly different. Students in the adaptive group (experimental group) were assigned to use the adaptive SQL learning system for two weeks whereas the control group used the traditional e-learning system. Students who already scored 100% in quiz #1 were not assigned to any group because we assumed that they have totally understood the basic SELECT command. The exercises in the system written in Thai were verified by IT-oriented experts. After ending two weeks, both groups took a basic SELECT command quiz (Quiz #2) having the same number of questions, knowledge, score, and difficulty as those of quiz #1. The study used scores from both quizzes to analyze the results. We assigned half of the students must use the system (#1), but optional for the other half (#2). We found that 42 of #1 students actually used the system and 38 of #2 students did not.

From quiz #1, the scores of the adaptive group and the control group were close: both at 60% – see Table 2.

To ensure that there is no difference in background of sampled groups, we checked that SQL knowledge for the two groups should be similar. So, a two-sample z-test evaluated the difference between two groups, based on these hypotheses:

H_0: $\mu_1 = \mu_2$ (Students in the adaptive group and students in the control group have the same level of SQL knowledge.)

H_1: $\mu_1 \neq \mu_2$ (Students in the adaptive group and students in the control group have a different level of SQL knowledge.)

With $\alpha = 0.01$ and a two-tail test, the critical values of z are -2.576 and $+2.576$. We accepted H_0, because $z = -0.5503$ lies in $(-2.576, 2.576)$. Thus, at $\alpha = 0.01$, $p < 0.010$, there was no significant difference in SQL knowledge between two groups.

TABLE 2. Mean scores for both quizzes.

Sample	No. of Sample	Quiz #1 Score		Quiz #2 Score		Score Changed		Score Increased (%)
		Mean (%)	S.D.	Mean (%)	S.D.	Mean (%)	S.D.	
Adaptive group	42	57	5.6	72	5.6	26	5.7	81
Control group	38	60	5.3	59	6.8	-1	6.6	58

B. LEARNING OUTCOME EVALUATION

After finishing the experiment, the mean scores from quizzes #1 and #2 were compared: the mean score of students in adaptive group increased from 14.3 to 18.1 (a 26% improvement). On the other hand, the control group score decreased by 1% – see Table 2.

The data in Table 2 show that 81% of students in the adaptive group scored better, with improvements greater than those in the control group. Only 58% of control group students improved their score, but 39% of those students actually lowered their score.

To show that students in the adaptive group had better learning outcomes, the study used quiz #2's scores on a two-sample z-test.

H_0: $\mu_1 > \mu_2$ (Students in the adaptive group had the better learning outcome)

H_1: $\mu_1 <= \mu_2$ (Students in the adaptive group did not have better learning outcome)

We found $z = 2.343$ (> -2.576, z of critical value at lower-tailed 0.005 alpha level). So, we accepted H_0: students in the adaptive group really had a better learning outcome.

C. KNOWLEDGE SUGGESTION EVALUATION

For knowledge suggestion evaluation, we analyzed the associated learning objectives of each question from quiz #2, in which each question assessed one of the learning objectives of SQL commands. If the student answered correctly two-thirds of all questions for each learning objective, we assumed that the student understood that associated learning objective. Table 3 shows the results.

TABLE 3. Knowledge suggestion evaluation.

# times using adaptive	Precision	Recall	F-Measure
1 time	81%	84%	82%
2 times	75%	93%	83%
>= 3 times	80%	99%	89%
OVERALL	77%	89%	84%

In the system, instructors can set achievement levels to match their requirements. In this test, we set 80% to represent satisfactory achievement for SQL knowledge in Thai universities and schools. Otherwise, the system will suggest students obtaining less than this level to practice more.

We found that knowledge suggestion precision – the ratio of true positive to both true and false positive values [26] – is 77%, recall is 89% and F-measure is 84%, or more than 80%, so we concluded that the result is satisfactory. False predictions arise from many false positives in integrated and complex questions. For example, one question might need an SQL answer with a combination of BETWEEN, LIKE and AND operators, but the student might not understand only the LIKE operator. This made the student answer this question incorrectly. Consequently, the knowledge suggestion result showed that the student did not understand all BETWEEN, LIKE and AND operators. This impacts knowledge suggestion results from other learning objectives (BETWEEN and AND operators) that they might have already understood. To prevent this false prediction, the grading system needs to be able to identify only the incorrect part of the query. This limitation of the grading system needs further work.

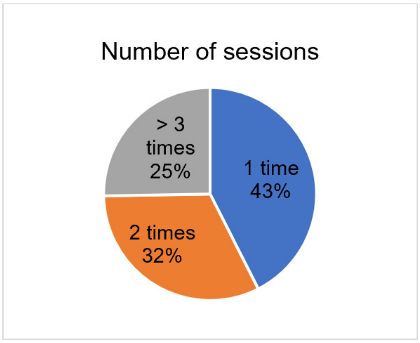


FIGURE 15. Number of sessions for Q1.

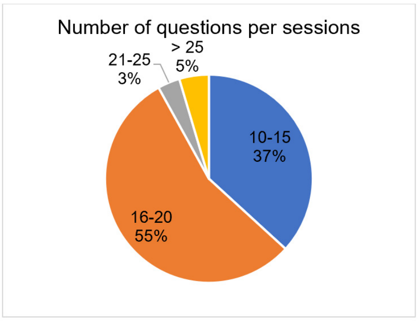


FIGURE 16. Q2: Number of questions per session.

D. USER'S PERSPECTIVE EVALUATION

In addition to evaluating by outcome-based learning, students were surveyed to evaluate system usability and student satisfaction with five questions scored on a 5-point Likert scale. There were 86 respondents from students in the adaptive group and students who received a full score in a preliminary test and voluntarily used the system. Results from each question are shown below.

Q1: How many sessions did you practice?

Fig. 15 shows that 43% of respondents only used the system once, but the others used the system more than one time.

Q2: How many questions did you practice per session on average?

Fig. 16 shows that most students (55%) chose to practice 16-20 questions. 37% of respondents attempted 10-15 questions per session. This data can be used to adjust the default question settings suggested for students in the future.

Q3: Before using the system, what level of understanding of SQL commands did you have?

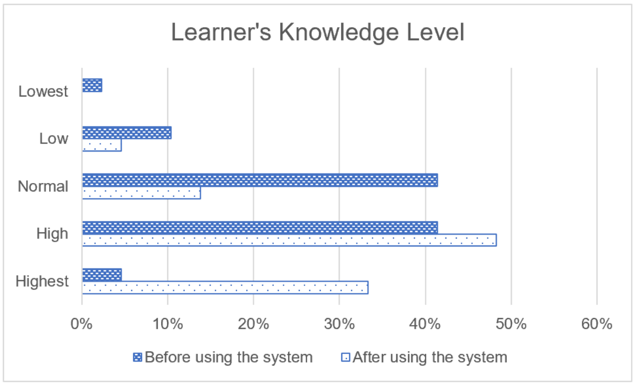


FIGURE 17. Q3 and Q4: knowledge level before and after using the system.

TABLE 4. Responses to question 5.

Question	Average	S.D.
Q5.1 The Adaptive SQL Learning system is easy to use.	3.6	0.8
Q5.2 The knowledge suggestion in the system suggests what to improve for student, which precisely directs to your weak point.	3.8	1.0
Q5.3 There is variety of questions in the system which helped you improve your SQL skill.	3.4	1.0

Q4: After using the system, what level of understanding of SQL commands did you have?

Fig. 17 (the highlighted bars) demonstrated that 88% of students evaluated their level of knowledge as medium (41%), high (42%) to very high (5%). 12% of students evaluated themselves as below medium level.

In Fig. 17, total number of students with good understanding, i.e. number at the high level and above, increased from 46% to 81% (the white bars) and is clearly greater than before taking our system, thus clearly shows that our system helped students improve their understanding of SQL command.

Q5: Do you agree with the following statements?

Table 4 shows that knowledge suggestion really facilitated learning and the system was easy to use. However, the diversity of questions was perceived as mediocre: this may be because students also learnt other more difficult commands in laboratory class while we ran this experiment. Thus students need more questions, which are linked to the same learning objectives as the one they are currently studying in the laboratory class, so we need to modify question settings to generate more diverse exercises.

VII. CONCLUSION

The study used reverse SQL question generation algorithm (RSQLG) to solve some difficulties in setting up adaptive e-learning systems, in particular suggestion of suitable questions for each student and insufficient numbers of exercises available to students. Our algorithm reverses the manual question writing process by starting with the answer. The algorithm used a database as input to generate the SQL query and, from the query, generated the text description. We suggested and delivered questions to students using our rule-based adaptation technique from data derived from previous answers. We evaluated our approach with 86 students, who enrolled in a database system concept course. Our adaptive e-learning system improved their skills and led to better learning outcome than from students who did not use it. The question adaptation module also produced a satisfactory prediction of student SQL knowledge level. Moreover, a survey on user satisfaction demonstrated that students agreed that the system facilitated their understanding. In summary, our main contributions are three-fold. First, our RSQLG algorithm succeeded in automatically generating basic SQL questions. Second, the adaptive e-learning framework for SQL practical course was novel. Third, the adaptive e-learning framework using RSQLG improved student learning outcomes.

In future work, we will enhance the RSQL generation algorithm with more exercise variety and complexity and, also analyzing fields and table text descriptions without manual input to database metadata. The complexity in SQL exercise will include GROUP BY and JOIN clauses. We can also improve the text explanation to support other languages with more complex grammar.

REFERENCES

[1] R. D. Ray, "Adaptive computerized educational systems: A case study," in *Evidence-Based Educational Methods*, D. Moran and R. Mallott, Eds. Amsterdam, The Netherlands: Elsevier, 2004, pp. 143–170.

[2] C. Domínguez and A. Jaime, "Database design learning: A project-based approach organized through a course management system," *Comput. Educ.*, vol. 55, no. 3, pp. 1312–1320, Nov. 2010.

[3] M. Cvetanovic, Z. Radivojevic, V. Blagojevic, and M. Bojovic, "ADVICE—Educational system for teaching database courses," *IEEE Trans. Educ.*, vol. 54, no. 3, pp. 398–409, Aug. 2011.

[4] P. Brusilovsky and C. Peylo, "Adaptive and intelligent Web-based educational systems," *Int. J. Artif. Intell. Edu.*, vol. 13, nos. 2–4, pp. 159–172, 2003.

[5] E. J. Brown, T. J. Brailsford, T. Fisher, and A. Moore, "Evaluating learning style personalization in adaptive systems: Quantitative methods and approaches," *IEEE Trans. Learn. Technol.*, vol. 2, no. 1, pp. 10–22, Jan. 2009.

[6] *The VARK Modalities*. Accessed: Aug. 1, 2018. [Online]. Available: http://vark-learn.com/introduction-to-vark/the-vark-modalities/

[7] A. Y. Kolb and D. A. Kolb, *Learning Style Inventory: Technical Manual*. Boston, MA, USA: McBer & Co, 1976.

[8] *MBTI Basics*. Accessed: Aug. 2, 2018. [Online]. Available: http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/

[9] R. M. Felder and L. K. Silverman, "Learning and teaching styles in engineering education," *Eng. Educ.*, vol. 78, pp. 674–681, 1988.

[10] I. A. Alshalabi, S. Hamada, and K. Elleithy, "Automated adaptive learning using smart shortest path algorithm for course units," in *Proc. Long Island Syst., Appl. Technol.*, May 2015, pp. 1–5.

[11] N. Zemirline, Y. Bourda, and C. Reynaud, "Expressing adaptation strategies using adaptation patterns," *IEEE Trans. Learn. Technol.*, vol. 5, no. 1, pp. 38–51, 1st Quart., 2012.

[12] M. Jadhav, S. Rizwan, and A. Nehete, "User profiling based adaptive test generation and assessment in E—Learning system," in *Proc. 3rd IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2013, pp. 1425–1430.

[13] H. Wainer, N. J. Dorans, R. Flaugher, B. F. Green, and R. J. Mislevy, *Computerized Adaptive Testing: A Primer*. Evanston, IL, USA: Routledge, 2000.

[14] C. Pahl and C. Kenny, "Interactive correction and recommendation for computer language learning and training," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 6, pp. 854–866, Jun. 2009.

[15] A. Mitrovic, "An intelligent SQL tutor on the Web," *Int. J. Artif. Intell. Educ.*, vol. 13, nos. 2–4, pp. 173–197, Apr. 2003.

[16] S. Nalintippayawong, K. Atchariyachanvanich, and T. Julavanich, "DBLearn: Adaptive e-learning for practical database course—An integrated architecture approach," in *Proc. 18th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jun. 2017, pp. 109–114.

[17] K. Lee, S.-O. Kweon, H. Seo, and G. G. Lee, "Generating grammar questions using corpus data in L2 learning," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2012, pp. 443–448.

[18] N. Funabiki, T. Mohri, and S. Yamaguchi, "Toward personalized learning in JPLAS: Generating and scoring functions for debugging questions," in *Proc. IEEE 5th Global Conf. Consum. Electron.*, Oct. 2016, pp. 1–4.

[19] S. Yamaguchi, T. Mohri, and N. Funabiki, "A function for generating debugging questions in a Java programming learning assistant system," in *Proc. IEEE 4th Global Conf. Consum. Electron. (GCCE)*, Oct. 2015, pp. 350–353.

[20] K. Nandhini and S. R. Balasundaram, "Math word question generation for training the students with learning difficulties," in *Proc. Int. Conf. Workshop Emerg. Trends Technol.*, Feb. 2011, pp. 206–211.

[21] M. Liu, R. A. Calvo, A. Aditomo, and L. A. Pizzato, "Using wikipedia and conceptual graph structures to generate questions for academic writing support," *IEEE Trans. Learn. Technol.*, vol. 5, no. 3, pp. 251–263, Jul./Sep. 2012.

[22] U. Jain, Z. Zhang, and A. Schwing, "Creativity: Generating diverse questions using variational autoencoders," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6485–6494.

[23] S. A. Khalek and S. Khurshid, "Automated SQL query generation for systematic testing of database engines," in *Proc. IEEE/ACM Int. Conf. Automated Softw. Eng.*, Sep. 2010, pp. 329–332.

[24] K. Atchariyachanvanich, S. Nalintippayawong, and T. Permpool, "Development of a MySQL sandbox for processing SQL statements: Case of DML and DDL statements," in *Proc. 14th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2017, pp. 1–6.

[25] B.-G. Itzik, L. Kollár, D. Sarka, and S. Kass, *Inside Microsoft SQL Server 2008-T-SQL Querying*. Redmond, WA, USA: Microsoft Pr., 2009.

[26] J. Han, J. Pei, and M. Kamber, "Model evaluation and selection," in *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011, pp. 364–370.

KANOKWAN ATCHARIYACHANVANICH received the B.Sc. degree in information technology from Assumption University, the M.S. degree in information management from the Asian Institute of Technology, the M.P.A. degree in international development from Tsinghua University, and the Ph.D. degree in informatics from the Graduate University for Advanced Studies, Japan. She is currently an Assistant Professor with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Thailand. Her prior research has been published in international journals and international conferences, such as ACM SIGecom Exchanges, the *International Journal of Electronic Customer Relationship Management*, *E-business and Telecommunications*, the International Conference on Electronic Commerce, and the International Conference on Computer and Information Science. Her research interests include e-learning, information technology adoption, e-business management, and consumer behavior in the digital market.

SRINUAL NALINTIPPAYAWONG received the B.B.A. degree in finance and the M.B.A. degree in management from Kasetsart University and the M.S. degree in information system management from the National Institute of Development Administration. She is currently a full-time Lecturer with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Thailand. Her research has been published in national and international conferences, such as the National Conference on Information Technology, the International Conference on Computer and Information Science, the International Joint Conference on Computer Science and Software Engineering, and the International Conference on Information Technology and Electrical Engineering. Her research interests include e-learning, strategic information technology management, intelligent systems, and teach startup.

THANAKRIT JULAVANICH received the B.S. degree in information technology from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 2017, where he is currently pursuing the M.S. degree in information technology. His research interests include technology-enhanced learning, intelligent systems, information systems, and web-based applications.

• • •