Automatic Generation of SQL Queries

Ms. Quan Do, New Mexico State University

Quan Do is currently pursuing her Ph.D. degree in Computer Science at New Mexico State University, Las Cruces, New Mexico. She received her M.S. In Information Systems and M.S. In Health Informatics from Marshall University, Huntington, WV. Her research interests are in automated SQL query generation, verification and validation of SQL queries, and Big Data Analytics.

Dr. Rajeev K Agrawal, North Carolina A&T State University

Dr. Rajeev Agrawal is an assistant professor in the department of computer systems technology at North Carolina A&T State University. He has published more than 40 referred journal and conference papers, and 4 book chapters. His current research focuses on Anomaly Detection in Computer Network, Big data Analytics, and Content-based Image Retrieval. He has also worked at HP Company in transportation, Medicaid Management Information System (MMIS) domains.

Dr. Dhana Rao

Dhana Rao is an Assistant Professor in Microbiology at Marshall University, West Virginia. She obtained her PhD in 2006 from the University of New South Wales, Australia. Her research interest are in metagenomics and bioinformatics

Prof. Venkat N Gudivada, Marshall University

Venkat N Gudivada is a Professor and interim Chair of the of the Weisberg Division of Computer Science at Marshall University. He received his Ph.D. in Computer Science from the Center for Advanced Computer Studies, University of Louisiana at Lafayette. encompass Big Data Analytics, Verification and Validation of SQL Queries, HPC-driven applications, and Personalized eLearning.

His prior experience include work as a Vice President for Wall Street companies in New York City for over six years including Merrill Lynch and Financial Technologies International. Previous academic tenure include work at the University of Michigan, University of Missouri, and Ohio University.

Automatic Generation of SQL Queries

Abstract

Structured Query Language (SQL) is an ANSI and ISO standard declarative query language for querying and manipulating relational databases. It is easy to write SQL queries but very difficult to validate them. Often students conclude that a SQL query is correct simply because the query compiles, executes, and fetches data. Therefore, it is crucial that SQL assessment tasks are carefully designed and implemented to ensure a deep learning experience for students. In this paper, we propose an approach to automatically generate SQL queries for assessing students' SQL learning. SQL concepts are modeled using RDFS. The user can select SQL concepts to be included in an assessment and our approach will generate appropriate queries. The proposed approach is generic and is database metadata driven. A Web-based prototype system is developed to illustrate the effectiveness of the proposed approach.

1. Introduction and Motivation

Automatic question generation is an active area of research. Its practical applications are many, especially in the context of learning assessment. In both online and traditional face-to-face courses, *learning outcomes* are used as standards for measuring and comparing performance and achievement of learners. In a broader context, *outcomes-based assessment* is widely used in evaluating and improving academic degree programs. The former plays a central role especially in ABET accreditation of undergraduate engineering and computer science degree programs.[1] Taxonomies such as Bloom's in the cognitive domain and SOLO are used to provide a shared language for describing learning outcomes and performance in assessments.[7]

Contrary to popular belief, online courses require more instructor time relative to face-to-face courses.[8] Factors such as providing immediate and frequent feedback and assessing expected outcomes objectively assume even greater importance in online courses. In this context, automatically generating questions that assess a given set of concepts at various levels of revised Bloom's taxonomy is a tremendous help for administering both online and face-to-face courses. This is the driver for our interest in automatic question generation in highly structured contexts.

In this paper, we address the problem of automatic generation of Structured Query Language (SQL) queries. SQL is a database language for querying and manipulating relational databases. Writing and executing SQL queries is an integral part of relational database courses. The latter is a required course in most computer science, information technology, and management information systems degree programs. SQL is standardized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO).

Since SQL is a *declarative query language*, it is easy to write SQL queries. A user simply specifies what data is to be retrieved rather than specifying how to retrieve the data. As a consequence, it is quite difficult to validate that a SQL query does what the specification has

called for. Often students conclude that a SQL query is correct simply because the query compiles, executes, and fetches data. Since typical databases are vast in size and data relationships are complex, it is not feasible to manually validate SQL queries by browsing the data. Therefore, it is crucial that SQL assessment tasks are carefully designed and implemented to ensure a deep learning experience for the students.

Our approach to automatic generation of SQL queries is shaped exclusively by the following considerations. Generating good questions manually is an intellectual activity and takes time. A good assessment should test for depth of understanding of the learner at various levels (e.g., six levels in the revised Bloom's taxonomy). It should be possible to validate the answers automatically. Especially for complex SQL queries, there can be multiple correct ways to write the query. Under this scenario, grading SQL questions can take substantial amount of instructor's time. One practical option is to compare the result of executing the SQL query (corresponding to the instructor's solution) with that of the result of executing the SQL query provided by the student. Though this provides some assurance about the correctness of student's SQL query, but there is no absolute guarantee since the results may differ for a different database state.

Using question banks for SQL assessment have their limitations. There are only a fixed number of questions and they will become public knowledge after a few semesters of usage. Furthermore, typically question banks are tied to a specific textbook. More importantly, test bank questions may not assess the SQL topics that an instructor has emphasized in the class. Also, test banks may not assess the topics at desired levels of the revised Bloom's taxonomy.

The overarching goal for our work is driven by the following considerations: questions should be generated on the fly, no question bank should be involved; answers to the questions should also be generated automatically; generated questions should feature all ANSI/ISO SQL concepts, individually or in permissible combinations; question generation system should work with any relational database created using any open source or commercial database management systems (that adhere to ANSI and ISO SQL standards); in addition to the generated questions and solutions, the system should provide additional information to facilitate **self-directed guided learning**.

Answering questions generated using the proposed approach requires deep understanding of SQL concepts. At the current stage of the work described in this paper, generated questions are meant for student assessment. These questions can be used for two distinct scenarios. In the first, students are asked to compute an answer for the query by manually executing it on a small database. Instructor obtains a solution for the same query by simply executing the query on the same database. Assessment is manually done, though it can be easily automated. In the second scenario, students are asked to describe in plain English what problem is solved by a given SQL query. Currently our system does not provide solutions to these queries. Also, assessment is completely manual for this category. Since our system does not generate multiple choice questions, generating distractors is not an issue.

The remainder of this paper is structured as follows. Related work is described in section 2. We describe our approach to automatic generation of SQL queries in section 3. Finally, section 4

indicates future extensions of this work and concludes the paper.

2. Related Work

In a broader context, there are two major approaches to automatic question generation. The first approach deals with generating questions from natural language texts. Currently there is a tremendous interest in this direction.[21,25,26] A recent edition of Dialog & Discourse journal features a special issue on automated question generation from natural language texts.[22]

The second approach deals with generating data and questions in a more structured and constrained contexts. Our approach to question generation presented in this paper falls under this category. Generating data and questions automatically that satisfy a given set of constraints has been actively investigated in diverse contexts.[10–14] A taxonomy of questions for question generation is presented in.[16] In the following, we provide a brief description of approaches to automated question generation. Given the diversity in context, purpose, and approach, it is not possible to group the approaches into semantically meaningful categories. These approaches are motivated by very specific practical needs and tend to be tactical rather than formulating a generic framework for question and answer generation. Also, information about their effectiveness evaluation experiments is not available in most cases.

An automatic question pattern generation method based on domain ontologies is discussed in.[18] A set of question patterns referred to as *predictive questions* are generated. *Query templates* provide answers to these questions. Answers to predictive questions are extracted from a knowledge base. This approach is motivated by the need for answering questions posed in natural language against a knowledge base.

Papasalouros et al.[19] describe an approach and a prototype implementation for automatically generating multiple choice questions using domain-specific ontologies. Ontologies are represented using OWL (Ontology Web Language).

Siddiqi et al.[27] describe IndusMarker, an automated short-answer marking system for improving teaching and learning of an object-oriented programming course. IndusMarker is designed for factual answers and is based on structure matching.

Al-Yahya[2] describes OntoQue system for generating questions for objective assessment. OntoQue uses knowledge inherent in domain ontologies to generate semantically correct assessment items.

A template-based approach to question generation is presented in.[10] A template is associated with each learning objective. The system is demonstrated for generating questions for tree data structures assessment.

Question generation component of DynaLearn is presented in.[13] It generates 23 different question types using a four-step question generation process.

Li and Sambasivam[11] describe a system for tutoring computer architecture and an approach to objectively assessing difficulty level of questions. The generated questions involve only one variable. In,[12] they extend the work to generate questions that involve multiple variables.

In software testing, *assertions* are used to uncover software defects. An assertion is a statement that must be true. An approach to automatically generating assertions to speedup assertion-based software testing is discussed in.[3]

Relational databases is an area that requires automatic generation of representative data to populate databases so that *applications* can be tested with realistic data.[6] Typically real data is not available in development and test environments due to security and privacy concerns. Automatic generation of SQL queries required for testing *relational database engines* is described in.[4,9,15,17]

Prior[24] examines the difficulty of assessing students' learning in formulating SQL queries. This study concludes that the conventional approach to SQL assessment — a written test where students write SQL queries, but the queries are not executed on a database — does little to encourage students to adopt a deeper learning approach to the subject. AsseSQL – an online environment for SQL assessment – is described in this paper. AsseSQL allows students to write and execute their queries and compare the results obtained with expected results. Questions are drawn from a question bank.

Brusilovsky et al.[5] describe *SQL Exploratorium* — an architecture for integrating tools for SQL learning. It offers three functions: a large set of interactive annotated SQL examples; SQL Knowledge Tester (SQL-KnoT), using which students can practice and test their SQL problem-solving skills; SQL-Lab, which allows students to formulate and execute queries, observe their results, and test performance of SQL scripts. Annotated examples explain what each line in a query does. SQL-KnoT generates question text using templates for a set of predefined databases. Questions are evaluated automatically by comparing the results obtained by executing the user query and the pre-stored solution query.

3. Proposed Approach to Automatic Generation of SQL Queries

Central to our approach are two knowledge structures: a SQL ontology expressed as RDFS,[5] and metadata about databases. The SQL ontology is available for free download at.[20] It has 347 RDF triples and defines 166 rdfs:Class's. For the purpose of this paper, SQL ontology models domain concepts at the right level of granularity. The SQL ontology is the basis for selecting concepts for automatic question generation. A view of the ontology displayed in Protege is shown in figure 1.

ISO SQL standard specifies a requirement that all relational database management systems (DBMS) should provide access to database metadata. The latter comprises information such as names of databases, names of tables in a database, attributes in a given table, database constraints, and database indexes, among others. In PostgreSQL DBMS,[23] metadata information is stored in a database schema named *information_schema* and the latter has several base and virtual tables including pg_tables, pg_attribute, pg_constraints, pg_database, and pg_index, among others.

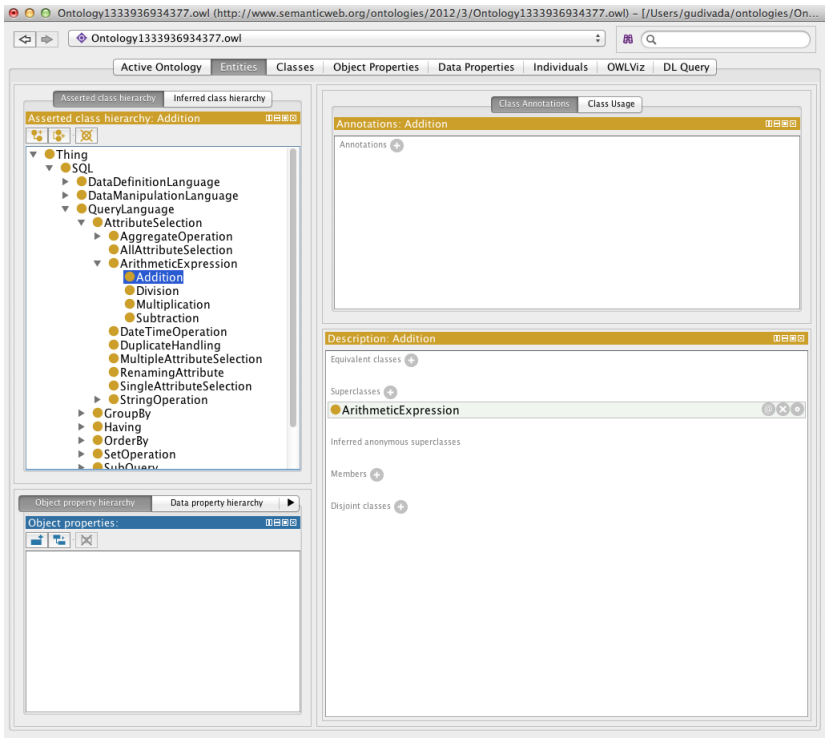

Figure 1: A view of the SQL ontology displayed in Protege

Therefore, metadata can easily be retrieved using the SQL language itself.

We have used Java 7, Tomcat, JDBC, JSPs and Servlets, HTML5, CSS3, PostgreSQL, and JavaScript for developing the Web application. There is no compelling reason to develop the solution as a Web application but for the accessibility convenience through a Web browser.

3.1 Automatic Generation of SQL Queries

The home page of the Web application for automatic generation of SQL queries is shown in figure 2. A user first selects a database against which to generate queries using the dropdown list at the center-top. Using the link immediately below one, the user can view the data model of the database. The left column radio buttons are used to indicate how many tables the query should involve (one, two, or more than two). Alternative, the user can indicate specific tables that should appear in the query using the list box right below the radio button group.

The SQL ontology is displayed in the form of a tree in the center scrollable list box. The user can check one or more SQL concepts to indicate which concepts should appear in the query. SQL query is generated by clicking on the button labeled *Generate Query*.

Successful generation of the query involves randomly selecting the required number of tables (if the user has not explicitly indicated table names) so that the generated query illustrates the chosen SQL concepts. The system also needs to deal with situations where the user has specified less

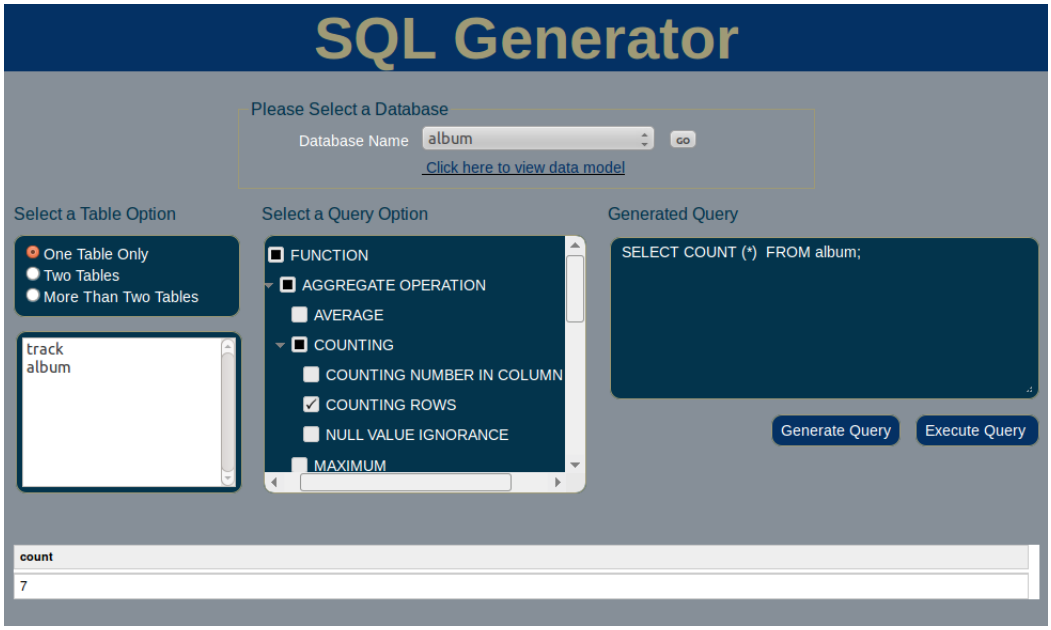

Figure 2: Web application for automatic SQL query generation

than the number of tables required to generate the query. For example, if the user has specified only one table and has selected **inner join** SQL concept, the system must determine the name of other table which has referential integrity relationship with the table specified by the user. The metadata table named pg_constraints is queried to determine the table name.

3.2 Automatic Generation of Solutions to SQL Queries

Since the Web application is connected to a PostgreSQL server, executing the generated SQL query is simply sending the query to the server using JDBC and displaying the results returned (see figure 4). These results are compared with those obtained by executing the SQL query written by the student. As noted earlier, multiple solutions can exist for complex SQL queries and this is an automated way to validate student queries very quickly.

3.3 SQL Questions Encompass all ANSI/ISO SQL Concepts

This feature primarily depends on the comprehensiveness of the SQL ontology. The mapping of user specified SQL concepts to question templates is stored in XML files. Replacing one SQL ontology with another requires changes only to this XML file, and no source code changes are required. The query generation templates construction takes into account the grammatical structure of SQL queries (figure 4). A basic SQL query can be constructed using only constructs in the first two lines. The remaining four clauses are optional. Not all the four need to be present (i.e., a subset is permitted). However, if a subset of them are present, they should appear in the order shown in figure 4.

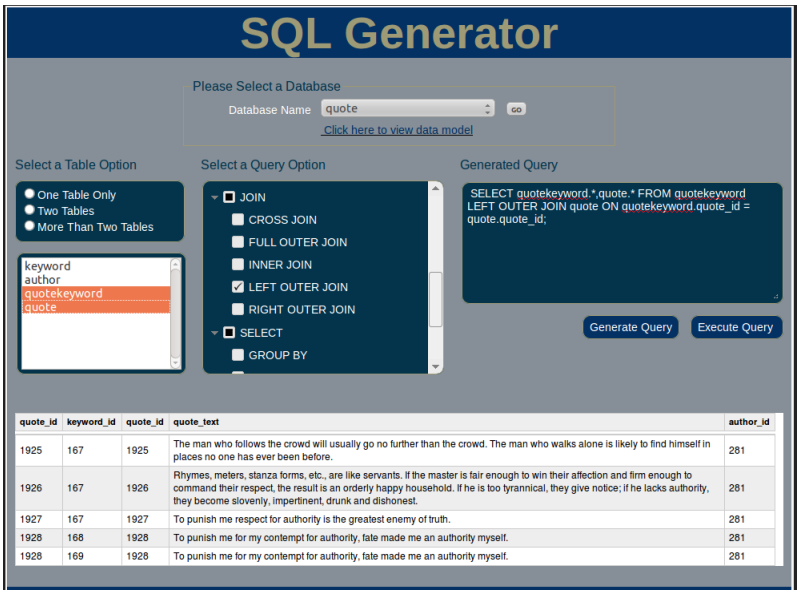

Figure 3: Query to illustrate SQL *LEFT OUTER JOIN* concept

```
SELECT    <column names>
FROM      <table names> and <join conditions>
WHERE     <row restrictions>
GROUP BY  <column names for grouping>
HAVING    <condition specifying which groups to keep>
ORDER BY  <sorting specification for displaying data>;
```

Figure 4: General structure of SQL queries

3.4 Dealing with Different Open-source and Commercial Databases

Though the ANSI/ISO SQL standard exist, not all DBMS vendor products comply with the standard. Typically, commercial vendors implement a subset of the SQL standard and also provide additional SQL features that are unique to their products. Open source DBMS products often implement only a subset of the SQL standard and provide no additional features unique to their products. DBMS vendors vary widely in how metadata is stored and managed. This makes it difficult for automated SQL generators to work with diverse open source and commercial DBMS products.

Our approach to this problem is to use a software design pattern called **strategy**. The latter allows for defining a family of algorithms, encapsulating each one, and making the algorithms interchangeable. We have defined a generic interface for metadata access to relational databases. An algorithm corresponds to providing a concrete implementation of this generic interface for a specific DBMS such as MySQL. Strategy design pattern lets these algorithm vary independently from the Web application for automatic query generation. This work is in progress.

3.5 Facilitating Self-directed Guided Learning

There are two scenarios for self-directed guided learning. In the first scenario, the system generates the query and asks the user to explain what the query does. In the second instance, the user is given the query in natural language and is asked to write the corresponding SQL query. In both cases, the user may request hints and the system responds appropriately so that the solution emerges incrementally.

Consider the first scenario. For any given SQL query, following are the standard set of hints:

1. Where does the data come from? That is, which tables do you need to access? (partial FROM clause).
2. Which columns do you need to retrieve? (the SELECT clause). If the data comes from more than one table, how do you associate the data in a table with its related data in another table? (specifying JOIN conditions)
3. How do you restrict rows from each table? In other words, how do you specify the WHERE clause?
4. Do you need to group data? If so, on what columns do you group data? In other words, how do you specify the GROUP BY clause?
5. Do you consider all groups? If so, what criteria do you use to determine which groups to keep and which groups to reject? In other words, how do you specify the HAVING clause?
6. Do you need to sort the data? On what columns? In which order? In other words, how do you specify the ORDER BY clause?

The second scenario is more involved. The system needs to generate textual description of the auto generated query. Once this is done, the second scenario is identical to the first. Generating textual description requires parsing the auto generated SQL query first to construct the corresponding Abstract Syntax Tree (AST). Text generation algorithms will be used to produce textual descriptions of SQL queries. This work is also under progress.

4. Conclusions and Future Work

In this paper, we have described a Web-based system for automatically generating SQL queries. Given the difficulties in validating SQL queries, it is important to ensure that students learn SQL concepts by using appropriately designed learning activities and assessments. The system described in this paper is our first step towards achieving this goal.

Our future research direction is multifold. First, we will implement the *strategy* design pattern so that our system can work with databases which are managed by MySQL, Oracle, and Microsoft

SQL Server. The second one involves generating natural language description of an auto generated SQL query using textual generation algorithms. Third, we plan to extend the approach to generate multiple choice questions. Fourth, we will measure the effect of the Web-based system on student learning by having control and experimental groups.

References

[1] ABET, Inc. `www.abet.org/index.aspx`, August 2013. Last checked: 5 August 2013.

[2] M. Al-Yahya. OntoQue: A question generation engine for educational assesment based on domain ontologies. In *Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on*, pages 393–395, July 2011.

[3] Ali M. Alakeel. A framework for concurrent assertion-based automated test data generation. *European Journal of Scientific Research*, 46(3):352 – 362, 2010.

[4] Carsten Binnig, Donald Kossmann, Eric Lo, and M. Tamer Özsu. QAGen: generating query-aware test databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 341–352, New York, NY, USA, 2007. ACM.

[5] Peter Brusilovsky, Sergey Sosnovsky, Danielle H. Lee, Michael Yudelson, Vladimir Zadorozhny, and Xin Zhou. An open integrated exploratorium for database courses. *SIGCSE Bull.*, 40(3):22–26, June 2008.

[6] David Chays. Test data generation for relational database applications. Technical Report TR-CIS-2005, Department of Computer and Information Science, Polytechnic University, Long Island, NY, 2005.

[7] Ursula Fuller, Colin G. Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L. Lewis, Donna McGee Thompson, Charles Riedesel, and Errol Thompson. Developing a computer science-specific learning taxonomy. *SIGCSE Bull.*, 39:152–170, December 2007.

[8] V. Gudivada, R. Agrawal, and C. Chu. Online teaching and learning strategies for programming-intensive courses. In *Proceedings of the 9^{th} International Conference on Information Technology - New Generations (ITNG)*, pages 781 – 782. Conference Publishing Services – CPS, April 2013.

[9] Abdul Shadi Khalek and Sarfraz Khurshid. Automated sql query generation for systematic testing of database engines. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 329–332, New York, NY, USA, 2010. ACM.

[10] M. Lemos, R. Muralidharan, and V. V. Kamat. Automatic generation of questions for on-line evaluation. `www.cdac.in/html/pdf/Session4.2.pdf`, 2011.

[11] Tao Li and Sam Sambasivam. Question difficulty assessment in intelligent tutor systems for computer architecture. In *Proc ISECON*, pages 1–8. EDSIG, 2003.

[12] Tao Li and Sam Sambasivam. Automatically generating questions in multiple variables for intelligent tutoring. In *Society for Information Technology & Teacher Education International Conference (SITE)*, pages 471 – 2005, 2005.

[13] Floris Linnebank, Jochem Liem, and Bert Bredeweg. Question generation and answering. Technical Report D3.3, University of Amsterdam, 2010.

[14] Horst Liske. A framework for automated generation of examination questions from web based semantically treated search results. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, CompSysTech '11, pages 510–515, New York, NY, USA, 2011. ACM.

[15] Chaitanya Mishra, Nick Koudas, and Calisto Zuzarte. Generating targeted queries for database testing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 499–510, New York, NY, USA, 2008. ACM.

[16] R. Nielsen, J. Buckingham, G. Knoll, B. Marsh, and L. Palen. A taxonomy of questions for question generation. In *Workshop on the Question Generation Shared Task and Evaluation Challenge*, 2008.

[17] Christopher Olston, Shubham Chopra, and Utkarsh Srivastava. Generating example data for dataflow programs. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 245–256, New York, NY, USA, 2009. ACM.

[18] Shiyan Ou, Constantin Orasan, Dalila Mekhaldi, Laura Hasler, and Laura Hasler. Automatic question pattern generation for ontology-based question answering. In *FLAIRS Conference*, pages 183 – 188, 2008.

[19] Andreas Papasalouros, Konstantinos Kanaris, and Konstantinos Kotis. Automatic generation of multiple choice questions from domain ontologies. In Miguel Baptista Nunes and Maggie McPherson, editors, *e-Learning*, pages 427 – 434. IADIS, 2008.

[20] PAWS. SQL Ontology. `http://www.sis.pitt.edu/~paws/ont/sql.rdfs`, April 2013.

[21] Paul Piwek and Kristy Elizabeth Boyer. The third workshop on question generation. `http://oro.open.ac.uk/22343/1/QG2010-Proceedings.pdf`, 2010.

[22] Paul Piwek and Kristy Elizabeth Boyer. Special issue on question generation. *Dialog & Discourse*, 3, 2012. `http://elanguage.net/journals/dad/issue/view/347`.

[23] PostgreSQL. PostgreSQL - Relational Database Management System. `http://www.postgresql.org/`, April 2013.

[24] Julia Coleman Prior. Online assessment of sql query formulation skills. In *Proceedings of the fifth Australasian conference on Computing education - Volume 20*, ACE '03, pages 247–256, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[25] Vasile Rus and Arthur Graesser, editors. *The Question Generation Shared Task and Evaluation Challenge*, February 2009.

[26] Vasile Rus and James Lester, editors. *Proceedings of the 2nd Workshop on Question Generation, AIED 2009*, 2009.

[27] Raheel Siddiqi, Christopher J. Harrison, and Rosheena Siddiqi. Improving teaching and learning through automated short-answer marking. *IEEE Transactions on Learning Technologies*, 3:237–249, 2010.