Wo steckt nur der Fehler in der SQL-Anfrage? Semantische Prüfung von Lösungen

Inga Marina Saatz[1]

Abstract: This article introduces the web-based learning environment SQLearn, which allows learners to receive automated, individual feedback on the SQL queries. The SQLearn system performs a syntax check as well as a check of the semantic and structure of the formulated SQL query. To give feedback about semantic errors, the parse tree of the SQL query is compared with the parse trees of the sample solutions. First results of the SQLearn learning environment in a university database course are presented.

Abstract: In diesem Beitrag wird die webbasierte Lernumgebung SQLearn vorgestellt, mit der Lerner automatisierte, individuelle Rückmeldungen zu den von ihnen formulierten SQL-Anfragen erhalten können. Die Rückmeldungen beziehen neben einer Syntaxprüfung und dem Ergebnismengenvergleich auch die Struktur der formulierten SQL-Anfrage bei der logischen Prüfung der Korrektheit mit ein. Zur Korrektheitsprüfung wird der zur SQL-Anfrage gehörende Parserbaum mit den Parserbäumen der Musterlösungen verglichen und Lösungshinweise zu semantischen Fehlern generiert. Erste Ergebnisse des Einsatzes der Lernumgebung in einer Datenbanken-Lehrveranstaltung werden vorgestellt.

Keywords: SQL, Grading, Korrektheitsprüfung, Evaluation

1 Einleitung

Um performante datenzentrierte Anwendungen zu programmieren, ist oftmals eine Optimierung von SQL-Anfragen durch eine Umformulierung der SQL-Anfrage unter Beibehaltung der semantischen Korrektheit erforderlich. Gerade das Formulieren von SQL-Anfragen bereitet einem Teil der Studierenden in einführenden Datenbankveranstaltungen Schwierigkeiten (vgl. beispielsweise [My07]). Hinzu kommt, dass durch das Datenbank-Management-System (DBMS) lediglich eine Fehlermeldung im Falle eines syntaktischen Fehlers zurückgegeben wird. Die semantische Korrektheit kann nur über einen Vergleich mit Musterlösungen erfolgen. Weiter erschwert wird dies durch die hohe Zahl strukturell unterschiedlicher, semantisch äquivalenter SQL-Anfragen.

In diesem Beitrag soll der Frage nachgegangen werden, wie Studierende automatisierte, individuelle Rückmeldungen zu den von ihnen formulierten SQL-Anfragen erhalten können, welche auch die Struktur der formulierten SQL-Anfrage berücksichtigen.

Zur Beantwortung dieser Frage wurde die webbasierte Lernumgebung SQLearn entwickelt, welches eine Syntaxprüfung und den Ergebnismengenvergleich durchführt sowie die Struktur der formulierten SQL-Anfrage mit der Struktur von in einem

[1] Fachhochschule Dortmund, Fachbereich Informatik, Emil-Figge Str. 42, 44227 Dortmund, inga.saatz@fh-dortmund.de

Lösungspool hinterlegten Musterlösungen vergleicht und daraus eine Rückmeldung generiert.

Dazu werden zunächst im Kapitel 2 Lösungsansätze zur Unterstützung des Erlernens der deklarativen Anfragesprache SQL betrachtet. In Kapitel 3 wird das SQLearn-System vorgestellt und erste Ergebnisse des Einsatzes des Lernsystems in einer Pilotphase in Kapitel 4 evaluiert. Im anschließenden Kapitel 5 erfolgt die Diskussion der Ergebnisse und ein Ausblick wird gegeben.

2 Grundlagen

Zentraler Inhalt der betrachteten Lehrveranstaltung „Datenbanken 1“ ist das Erlernen der deklarativen Datenbankanfragesprache SQL (Structured Query Language) und deren Anwendung auf konkrete Anwendungssaufgaben. Eine Voraussetzung für eine aktive Anwendung von SQL ist ein vorhandener Zugang zu einer Datenbank, der in den Praktikumsräumen bereitgestellt wird. Das konkrete Auffinden von semantischen Fehlern in syntaktisch korrekten SQL-Anfragen ist zeit- und betreuungsintensiv und stellt im Massenbetrieb eine Herausforderung sowohl an Lehrende als auch Studierende dar. Durch die sich hieraus ergebenden immanenten Wartezeiten können sich bereits in der Anfangsphase des Lernprozesses Motivationsverluste ergeben, die einer aktiven Auseinandersetzung mit den Lerninhalten entgegenwirken. Typische Herausforderungen für einen Teil der Studierenden treten bei dem Verständnis der knappen (englischen) Fehlermeldungen und dem Erkennen von semantischen Fehlern in den eigenen Lösungen auf. So wird beispielsweise nicht erkannt, dass eine Anfrage falsche oder eine andere Anzahl an Datensätzen als Ergebnis liefert. Um dies leichter zu erkennen, müssen die Ergebnismengen der SQL-Anfrage mit denen der Lösung verglichen werden. Ein System zur angemessenen Visualisierung von Ergebnismengen von SQL-Anfragen wird beispielsweise von [Ce11] vorgestellt. Dieser Ansatz wird durch das aSQLg-System [St13] verfeinert, welches beim Vergleich auch die Datentypen und die Sortierung der Ergebnismenge berücksichtigt.

Um eine automatisierte Überprüfung der semantischen Korrektheit von SQL-Anfragen zu erreichen, werden in webbasierten SQL-Lernprogrammen beispielsweise durch (z.B. [ed17, Ji10, Sa04, Ca15]) heuristische Verfahren verwendet, bei denen eine Lösung als korrekt gekennzeichnet wird, wenn die Ergebnismenge der Anfrage mit der Ergebnismenge der Lösung auf dem verwendeten (und ggf. auf weiteren) relationalen Datenbasen übereinstimmt. Die Autoren von [Sa04] geben beispielsweise an, dass die Gleichheit von SQL-Anfragen zu 95% erkannt wird. Neben dieser Gefahr von falsch-positiven Rückmeldungen bieten die Systeme von [ed17, Ji10, Sa04] keine automatisierte Möglichkeit, semantische Fehler zu identifizieren. Um eine weitergehende Identifizierung von semantischen Fehlern auf der Basis eines Ergebnismengenvergleichs zu erreichen, werden durch [Ca15] für mögliche fehlerhafte SQL-Anfragen spezielle Datenbestände automatisch generiert. Dieser Ansatz reduziert zwar auch den Anteil falsch-positiver Rückmeldungen, im Gegenzug ist allerdings eine erhöhte Anzahl von auszuführenden SQL-Anfragen und die Generierung einer Vielzahl von Datenbeständen erforderlich.

Automatisierte Strukturanalysen von SQL-Anfragen, welche auch die semantische Bedeutung mitberücksichtigen, wurden von [Do11, If14] vorgestellt. Bei beiden Ansätzen wird die SQL-Anfrage hierarchisch in einen Parserbaum zerlegt. Bei [If14] werden die einzelnen Komponenten der SQL-Anfrage dann mit Hilfe der Levenshtein-Distanz mit denen der Musterlösung verglichen und Unterschiede als Fehlermeldung angezeigt. Aufgrund der Verwendung der Levenshtein-Distanz können geringe Variationen, beispielsweise durch logische Umformungen oder der Nutzung von Tabellenalias, bereits zu falsch-positiven Rückmeldungen führen. Auch bei [Do11] wird die SQL-Anfrage in einen Parserbaum zerlegt und in der Form eines XML-Dokument erfasst, so dass neben dem Ergebnismengenvergleich auch ein Strukturvergleich durchgeführt werden konnte. Bei übereinstimmenden Ergebnismengen wird die studentische Lösung solange mit Hilfe von Transformationsregeln umgeformt, bis diese strukturell mit der hinterlegten Lösung übereinstimmt. Durch die Anwendung der vorgestellten Transformationsregeln kann beispielsweise eine SQL-Anfrage mit Unterabfrage in eine äquivalente Anfrage umgeformt werden, die stattdessen eine Verbundoperation nutzt. Eine Lösung wird als fehlerhaft gekennzeichnet, wenn sie nach Anwendung aller Transformationsregeln nicht mit der Musterlösung übereinstimmt. Allerdings decken die von [Do11] genutzten Transformationsregeln nicht alle möglichen SQL-Konstrukte ab und ein Bericht über den Einsatz des Gesamtsystems in der Lehre steht noch aus, so dass unklar bleibt, wie mit diesem System Lösungshinweise zu semantisch fehlerhaften Lösungen generiert werden können.

Die Arbeiten aus der Literatur zeigen Ansätze, bei der eine automatisierte Korrektheitsprüfung von SQL-Anfragen über den Vergleich von Ergebnismengen als auch von Parserbäumen erfolgt. Beide Ansätze werden nachfolgend durch das SQLearn-System kombiniert. Um den Berechnungsaufwands bei der Korrektheitsprüfung zu reduzieren, wird bei dem SQLearn-System ein Pool von Musterlösungen für Lösungen zu verschiedenen SQL-Dialekte hinterlegt. Durch den Vergleich mit den zugehörenden Parserbäumen der Musterlösungen kann zu einer semantisch fehlerhaften Lösung eine Musterlösung mit ähnlicher Struktur ermittelt werden. Dieser Strukturvergleich mit einer nahen Musterlösung eröffnet die Möglichkeit, den Lernern auch hinsichtlich der Struktur der formulierten SQL-Anfrage eine Rückmeldung geben zu können.

3 Das SQLearn-System

Um auch im Massenbetrieb der Hochschule eine individuelle Unterstützung bei dem Erlernen von SQL-Anfragen zu bieten, wurde die webbasierte Lernumgebung SQLearn als .NET-Anwendung entwickelt. Dieses nimmt die durch die Studierenden formulierten SQL-Anfragen entgegen und führt diese auf dem jeweils ausgewählten DBMS Oracle oder MySQL aus. Hierdurch wird eine einfache Gegenüberstellung der Anfragen in den unterschiedlichen SQL-Dialekten ermöglicht. Die Komplexität der SQL-Abfragen deckt die in dem Praktikum der Veranstaltung verwendeten Datenbankabfragen bis hin zu mehrfach verschachtelten Anfragen in der Auswahlbedingung ab.

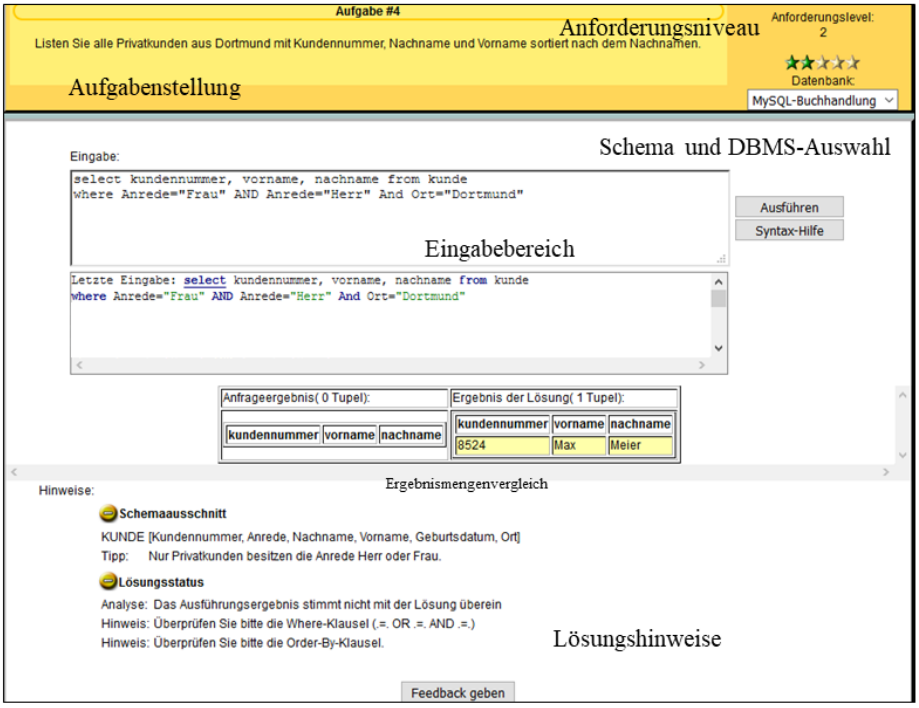

Abb. 1: Screenshot des SQLearn-Systems einer semantisch fehlerhaften SQL-Anfrage

Abbildung 1 zeigt eine beispielhafte Rückmeldung auf eine SQL-Anfrage. Zunächst wird durch einen Parser ein Syntaxcheck der eingegebenen SQL-Anfrage durchgeführt. Ist die Syntax der eingegebenen SQL-Anfrage fehlerhaft, so wird die Fehlermeldung des ausgewählten DBMS mit hinterlegten Kommentaren sowie ggf. vorhandenen deutschen Übersetzungen und zusätzlichen Hinweisen ergänzt. Nach [Mi00] zeigte die gleichzeitige Anzeige aller Fehlermeldungen den größten positiven Lerneffekt, daher werden auch durch das SQLearn-System alle verfügbaren Tipps und Lösungshinweise angezeigt (vgl. Hinweise in Abbildung 1).

Ist die SQL-Anfrage syntaktisch korrekt, dann kann sie durch das DBMS ausgeführt werden und im Rahmen des Ergebnisvergleichs wird die von dem DBMS zurückgelieferte Ergebnismenge mit der Ergebnismenge einer im Lösungspool vorhandenen Musterlösung verglichen. Dieser Vergleich wird in der Ergebnis-gegenüberstellung aufbereitet und dem Nutzer dargestellt.

Ein alleiniger Vergleich der Ergebnismengen reicht nicht aus, um eine falsch-positive Klassifizierungen der Lösung auszuschließen. Beispielsweise können zwei semantisch unterschiedliche Anfragen auf demselben Datenbestand jeweils eine leere Menge als Ergebnis liefern. Daher erfolgt neben dem Vergleich der Ergebnismengen zusätzlich ein Strukturvergleich der eingegebenen Lösung mit den im Lösungspool hinterlegten Musterlösungen.

Zur Durchführung des Strukturvergleichs wird die eingegebene SQL-Anweisung in einen Parserbaum umgewandelt. Zum Aufbau der Parserbäume werden die rein textuell vorliegenden SQL-Anfragen zunächst durch einen Tokenizer (basierend auf den SQL-Dialekten der beiden verwendeten DBMS) aufgeteilt und geparst. Durch den anschließenden Vergleich des generierten Parserbaums mit den Parserbäumen der Musterlösungen aus dem Lösungspool kann beispielsweise ermittelt werden, ob Schlüsselwörter, Tabellen oder Attributbezeichnungen fehlen. Um auch mehrfach verschachtelte SQL-Anfragen zu analysieren und vergleichen zu können, wurden der Parser, der Tokenizer und die Durchführung des Vergleichs der Parserbäume in der funktionalen Programmiersprache F# implementiert. Hierdurch ist es möglich, durch die Anwendung von funktionalen Mustervergleichen (pattern matching) beispielsweise eine Unterabfrage in der WHERE-Klausel zu parsen. Die Abbildung 2 zeigt den aufgebauten Parserbaum der Musterlösung zu der Aufgabe aus Abbildung 1. Die Punkte in dem Parserbaum stehen für Literale, z.B. der Ortsname „Dortmund“. Aus Gründen der Übersichtlichkeit sind die Literale in der Abbildung nicht aufgeführt. In der Abbildung sind diejenigen Klauseln fett markiert, in denen sich der Parserbaum der SQL-Anfrage von dem Parserbaum der Musterlösung unterscheidet. Diese Unterschiede werden gezählt und liefern so eine Distanzfunktion zum Vergleich von Parserbäumen, die sich von der Levenshtein-Distanz unterscheidet. Da auch die im Parserbaum enthaltenen Literale verglichen werden, besitzen beispielsweise die Ausdrücke COUNT(*) und COUNT(Kundennummer) eine Distanz von 1. Dies hat dann zur Konsequenz, dass für beide Ausdrücke jeweils eine eigene Musterlösung in den Pool aufgenommen werden muß, um beide Äquivalenzklassen von Lösungen abzudecken.

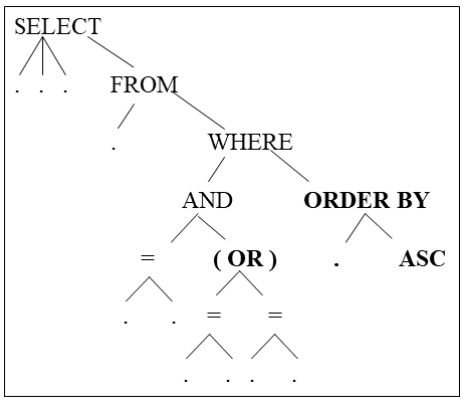

Abb. 2: Aufbau eines Parserbaums einer SQL-Anfrage

Der Strukturvergleich der SQL-Anfrage wird für alle Musterlösungen im Lösungspool durchgeführt. Für die Generierung der Lösungshinweise wird diejenige Musterlösung herangezogen, welche die geringste Anzahl von Strukturhinweisen liefert. Durch diese Heuristik soll erreicht werden, dass der Nutzer nur zu einer Musterlösung Strukturhinweise erhält und der Lerner zu derjenigen Musterlösung hingeleitet wird, die seinem eigenen Lösungsansatz am Nächsten kommt.

Damit dies auch in der Praxis realisiert werden kann, ist eine Vielzahl von Musterlösungen erforderlich. Beim Aufbau der Parserbäume werden bereits logische Äquivalenzumformungen (vgl. bspw. [He10]) programmseitig berücksichtigt, so dass soweit möglich von unterschiedlichen SQL-Dialekten, Unterschieden aufgrund der Verwendung von Alias und der Reihenfolge von Operanden abstrahiert werden kann. Hierdurch wird eine Reduktion der Menge der zu hinterlegenden Musterlösungen im Lösungspool ermöglich.

Prinzipiell können nicht alle Musterlösungen a priori im Lösungspool erfasst werden. Dabei ist allerdings zu berücksichtigen, dass bereits beispielsweise das Problems des Findens einer optimalen Join-Reihenfolge im Rahmen der Anfrageoptimierung des DBMS NP-hart ist [Ib84]. Es ist daher anzustreben möglichst viele, für die Lerner naheliegende Lösungen im Lösungspool zu ergänzen. Das Logfile wird dazu verwendet, um diejenigen Anfragen zu speichern, die eine korrekte Ergebnismenge liefern, jedoch nicht als Musterlösung hinterlegt sind oder durch den Parser auf eine Musterlösung zurückgeführt werden können. Durch die Anwendung von Transformationsregeln (s. bspw. [Do11]) können anschließend aus den vorhandenen Musterlösungen weitere äquivalente Lösungen generiert werden. Eine manuelle Erfassung und sukzessive Ergänzung des Lösungspools um weitere Lösungen während des laufenden Betriebs ist beim SQLearn-System vorhanden, da alle Aufgaben und Musterlösungen in einer relationalen Datenbank gespeichert werden.

4 Erste Erfahrungen

Das SQLearn-System wurde in der Veranstaltung „Datenbanken 1“ im Studiengang Medizinische Informatik im Wintersemester 2013/14 bei einem Pretest durch 32 Studierenden und im Wintersemester 2014/15 in einer Pilotphase durch 13 Studierenden genutzt. Aufgrund des Auslaufens der zugrundeliegenden Prüfungsordnung waren die Studierenden im Wintersemester 2014/15 allesamt Wiederholer aus höheren Semestern und die Kohorte im Vergleich zum Pretest auf etwa die Hälfte reduziert. Für die Veranstaltung im Wintersemester 2014/15 hatten sich 60 Studierende registriert, von denen letztendlich 17 an der Kursabschlussprüfung teilnahmen. Die Logfile-Analyse ergab, dass 6 von den Klausurteilnehmern das SQLearn-System genutzt hatten. Die Nutzung des SQLearn-Systems geschah auf freiwilliger Basis ohne zusätzliche Leistungsanreize und war eine Ergänzung zu dem Praktikum und den über das Learning Management System bereitgestellten Musterlösungen zu den Praktikumsaufgaben.

Es wurden durch das System 53 Aufgaben und insgesamt 89 Musterlösungen bereitgestellt. Abbildung 3 zeigt das Histogramm der Anzahl der Datenbankanfragen pro Semesterwoche. Dieses zeigt während der Behandlung von SQL-Abfragen während der Vorlesung und dem begleitenden Praktikum zwischen der 7. und 9. Semesterwoche einen Anstieg der Abfragezahlen. Kurz vor dem Termin der Klausur wurde das System vereinzelt genutzt, um die Musterlösungen zu den Aufgaben anzuzeigen, da deren Anzeige zu diesem Zeitpunkt auf Wunsch der Studierenden darin integriert wurde. Dieses erklärt den weiteren Anstieg der Abfragehäufigkeit vor der Prüfungsklausur in der 22. Semesterwoche.

Eine intensive Nutzung des Systems über mehrere Stunden hinweg mit einer schrittweisen Progression zu Lösungen konnten durch das Log bei den einzelnen Nutzern festgehalten werden (vgl. Abbildung 4). Von einigen Nutzern wurden direkt korrekte Lösungen eingegeben. Gespräche mit solchen leistungsstärkeren Studierenden ergaben, dass diese das SQLearn-System mehr zur Überprüfung ihrer unter Nutzung einer lokalen Datenbank entwickelten Lösungen nutzten.

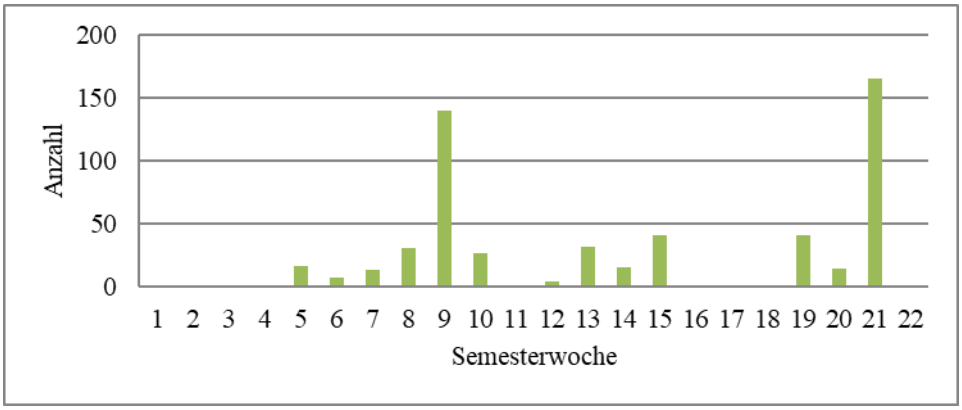

Abb. 3: Datenbankanfragen über das SQLearn-System pro Semesterwoche

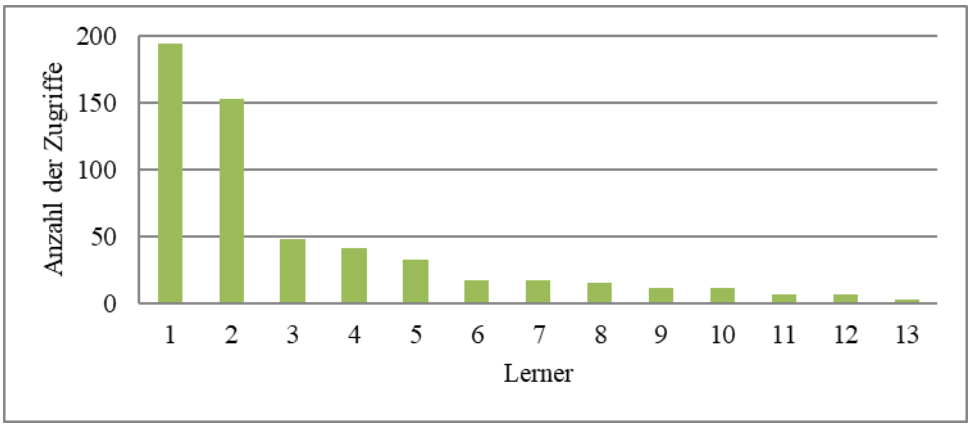

Abb. 4: Verteilung der Zugriffe auf das SQLearn-System auf die Nutzer

Bei der schriftlichen Evaluation durch einen Fragebogen zur Prüfungsvorbereitung im nachfolgenden Semester gaben 21 von 41 befragten Lernern an, dass sie mit dem SQLearn-System gearbeitet hatten. Die Rückmeldungen auf die offene Frage, weshalb sie mit dem System gearbeitet hatten, reichten von „gute Vorbereitung auf die Klausur“, „sehr hilfreich, solange man keine Software auf dem Rechner installiert hat“ und „hat mir sehr gut gefallen“ und „sehr gute Möglichkeit zum üben, mehr Befehle wären schön“. Dieses spricht dafür, dass aus Sicht der Studierenden die Verwendung des SQLearn-Systems auch einen Mehrwert im Vergleich zu den bereitgestellten Musterlösungen zu den Aufgaben bot. Einzelne Studierende merkten bei der Befragung

an, dass „manchmal Fehler“ auftraten. In Gesprächen mit einzelnen Nutzern stellte sich heraus, dass es als „Fehler“ wahrgenommen wurde, wenn eine korrekte Lösung durch das SQLearn-System nicht als „wahrscheinlich korrekte“ Lösung gekennzeichnet wurde. Dieses war der Fall, wenn die eingegebene SQL-Anfrage noch nicht als Musterlösung im SQLearn-System hinterlegt war. Aufgrund der relativ geringen Anzahl von Musterlösungen im Lösungspool ist diese Rückmeldung nachvollziehbar. Dieses zeigt, dass die Qualität der generierten Lösungshinweise noch weiter optimiert werden kann.

5 Diskussion und Ausblick

Der in diesem Beitrag vorgestellte Lösungsansatz zielt auf die Unterstützung der Studierenden im Lernprozess im Massenbetrieb der Hochschulen und dient nicht der automatisierten Lernerfolgsüberprüfung. Dementsprechend besitzt das SQLearn-System den Charakter eines Zusatzangebotes.

Es bleibt zukünftig zu untersuchen, inwieweit in Massenveranstaltungen weitere Nutzergruppen, insbesondere leistungsschwächere Studierende, erreicht und zu einer aktiven Auseinandersetzung mit den Lerninhalten motiviert werden können. Um dies zu erreichen, werden eine Erweiterung auf die Unterstützung u.a. DML-Operationen sowie ein Gamification-Ansatz weiterverfolgt. Perspektivisch wäre zudem eine Erweiterung des Anwendungsgebiets auf den Bereich des SQL-Tunings möglich. Die hierfür erforderliche weitergehende Erweiterung des SQLearn-Systems auf komplexere SQL-Anfragen, die bspw. Mengenoperationen beinhalten, oder weitere SQL-Dialekte weiterer relationaler DBMS ist aufgrund der Verwendung der funktionalen Programmiersprache durch eine Modifikation des Tokenizers und des Parsers möglich.

Der vorgestellte Lösungsansatz ist nicht auf deklarative Anfragesprachen beschränkt. Durch entsprechende Anpassungen, hauptsächlich des Parsers und des Tokenizers, konnte exemplarisch eine Aufgabe zur Erstellung einer objekt-orientierte LINQ-Anfrage über das SQLearn-System bereitgestellt werden.

Zusammenfassend wurde in diesem Beitrag das webbasierte Lernsystem SQLearn vorgestellt, mit dem Studierende automatisierte, individuelle Rückmeldungen zu den von ihnen formulierten SQL-Anfragen erhalten. Die Rückmeldungen beziehen die Struktur der formulierten SQL-Anfrage mit ein, so dass eine schrittweise Annäherung an die Musterlösung als auch kontextsensitive Lösungshinweise gegeben werden können.

Danksagung: Bedanken möchte sich die Autorin bei Sergej Savchenko für seine Unterstützung bei der Implementierung des SQLearn-Systems.

Literaturverzeichnis

[Ca15] Chandra, B.; Chawda, B.; Kar, B.; Reddy, K.V. M.; Shah, S.; Suarshan, S.: Data Generation for Testing and Grading SQL Queries. The VLDB Journal 24 (6), S. 731-755, 2015.

[Ce11] Cembalo, M.; De Santis, A.; Umberto, F.: SAVI: A new System for Advanced SQL Visualisation. In (Goda, B.; Sobiesk, E.; Connolly, R., Eds.): Proc. SIGITE'11, West Point, USA, p. 165-170, 2011.

[Do11] Dollinger, R.; Melville, N., Semantic Evaluation of SQL-Queries. In (Nedevschi, S., Eds.): Proc. Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference, 57-64, 2011.

[ed17] edb - Das eLearning Datenbank Portal, http://edb.gm.fh-koeln.de/, Stand: 26.08.2017

[He10] He-Biao, Y.; Li, C.; Fi-Fan, Y.: Study on the Static Analysis and the Similarity Comparing of SQL Code. In (Desheng, W.; Ruofeng, W.; Yi, X. Eds.): 3rd International Conference on Advanced Computer Theory and Engineering, p. 138-141, 2010.

[If14] Ifland, M.; Jedich, M.; Schneider, C.; Puppe, F. (2014): ÜPS – Ein autorenfreundliches Trainingssystem für SQL-Anfragen. In (Trahasch, S.; Plötzner, R.; Schneider, G.; Gayer, C.; Sassiat, D.; Wöhrle, N. Hrsg): Tagungsband der 12. E-Learning Fachtagung der Gesellschaft für Informatik, S. 259-264, 2014.

[Ib84] Ibaraki, T; Kameda, T.: Optimal nesting for comuting N-relational joins. ACM Trans. On Database Systems, 9 (3), p. 482-502, 1984.

[Ji10] Jian-gong, S.: The Design of Online Database Experiment System. In (Mahadevan, V.; Tomar, G. Eds.): Proc. 2nd International Conference on Education Technology and Computer (ICETC), p. 413-417, 2010.

[My07] Myers, C.; Douglas, P. (2007): The Un-Structured Student. In IEEE: Proc. 24th British National Conference on Databases (BNCOD'07), Glasgow, UK, p. 3-9, 2007.

[Mi00] Mitrovic, A.; Martin, B.: Evaluating the Effectiveness of Feedback in SQL-Tutor. In (Kinshuk, J.; Okamoto, T. Eds.) Proc. International Workshop on Advanced Learning Technologies, 2000. IWALT 2000, 143-144, 2000.

[St13] Stöcker, A.; Becker, S.; Garmann, R.; Heine, F.; Kleiner, C.; Bott, O.: Evaluation automatisierter Programmbewertung bei der Vermittlung der Sprachen Java und SQL mit den Gradern "aSQLg" und "Graja" aus studentischer Perspektive. In (Reiter, A; Rensing, C. Hrsg): Tagungsband der 11. E-Learning Fachtagung der Gesellschaft für Informatik, 233-238, 2013.

[Sa04] Sadiq, S.; Orlowska, M.; Sadiq, W.; Lin, J.: SQLator – An Online SQL Learning Workbench. In (Boyle, R.; Clark, M.; Kumar, A. Eds.): Proc. ITiCSE'04, Leeds, UK, 2004.