Generierung regulärer Ausdrücke zur Prüfung von SQL-Statements

Maximilian Förster[1], Patrick Binkert[2], Holger Rohland[3] und Thiemo Leonhardt[4]

Abstract: SQL-Ausdrücke sind durch die zugrundeliegenden Grammatiken in einer vielfältigen Art und Weise (optionale Leerzeichen, Umformungen) äquivalent formulierbar. Dies führt in e-Assessment-Szenarien zu Problemen bei der automatisierten Bewertung von SQL-Aufgaben. Dieses Problem wird bereits durch zahlreiche Ansätze auf vielfältigste Weisen gelöst. Allen diesen Ansätzen ist gemein, dass sie mit Umsetzungshürden für Lehrkräfte verbunden sind. In diesem Beitrag wird ein Ansatz über die automatisierte Generierung von Regulären Ausdrücken zur Lösung des Problems gewählt. Somit können vorhandene Fragestellungen ohne Übergangshürde in bereits bekannte digitale Formate überführt werden. Zur Evaluation des Ansatzes wurde ein Tool erstellt, das bei Eingabe einer Musterlösung als SQL-Ausdruck einen regulären Ausdruck automatisch generiert. Da die Bewertung einer korrekten Lösung an die Anforderungen des Lernsettings angepasst werden muss, bietet das Tool, die Möglichkeit den abgedeckten korrekten Lösungsraum einzuschränken. Die Evaluation teilt sich auf in das Testen der Funktionalität des Tools mit Probanden (99 % korrekt validiert) sowie auf die Akzeptanz des Ansatzes bei der Zielgruppe der Lehrpersonen über Interviews.

Keywords: e-Assessment, SQL, regulärer Ausdruck, Validierung

1 Motivation

Das Implementieren von Datenbankabfragen in SQL ist ein fester Bestandteil der schulischen und der universitären Lehre in der Informatik. Demzufolge existieren schon unterschiedliche Studien, die das Lehren und Lernen digital unterstützen sollen. Bei der Betrachtung dieser Tools [Na22] in Bezug auf die automatisierte SQL-Statement-Bewertung lassen sich für den einfachen und schnellen Einsatz im Kontext Schule drei Problemgruppen identifizieren: Gruppe 1: externe Tools (Active SQL, SQLator, AsseSQL), Gruppe 2: Konfigurations- und / oder Resourcenaufwand (SQL-Tester, SQL Knowledge Tester) und Gruppe 3: Verfahren mit (Teil-) CMS-Integration über Plug-Ins bzw. Schnittstellen, z.T. mit Notwendigkeit von Programmierkenntnissen je nach verwendeten LMS (aSQLg).

Ziel unseres Ansatzes ist es bei Lehrkräften bereits etablierter Learning-Managements-Systeme wie z.B. Moodle zu nutzen und gleichzeitig vorhandene Aufgabenformate leicht

[1] m.foerster@gts.lernsax.de
[2] patrick.binkert@cottygym.lernsax.de
[3] TU Dresden, Professur für Didaktik der Informatik, holger.rohland@tu-dresden.de
[4] TU Dresden, Professur für Didaktik der Informatik, thiemo.leonhardt@tu-dresden.de

doi: 10.18420/wsdelfi2023-31

überführbar zu machen, um vor allem im schulischen Bereich, in dem Datenbankabfragen nur einen vergleichsweise kleinen Lehr-Anteil haben, den Overhead für automatisierte Überprüfung so gering wie möglich zu halten. Unser stand-alone Online-Tool unterstützt die Lehrenden durch die Generierung lösungsäquivalenter regulärer Ausdrücke zu gegebenen SQL-Statements zur Überprüfung und Bewertung von Lösungsabgaben. Es ist ohne zusätzliche technische Kenntnisse und ohne zusätzlichen Ressourcenaufwand direkt einsetzbar oder bei Bedarf durch die Umsetzung als eigener Microservice in weitere Softwaresysteme integrierbar.

2 Generierung regulärer Ausdrücke aus SQL-Statements

Die strukturierte Abfragesprache SQL (structured query language) ist über Grammatiken genau spezifiziert, damit sind unterschiedliche Ausdrücke mit einem fachlich äquivalenten Ergebnis möglich. Als fachlich äquivalent bezeichnen wir eine Ausgabe der richtigen Attribute der entsprechenden Tupel unter Missachtung der Reihenfolge der Attribute. Gegebenenfalls kann dieses Spektrum durch den Nutzenden auch um Rechtschreibfehler im Statement ergänzt werden, wenn dies im Sinne des Lehrenden noch als fachlich äquivalent anzunehmen ist. Mit der Herausforderung diese fachlichen Äquivalenzen abzubilden, sehen sich Lehrende und Lernende in der Schule, der Hochschulbildung und anderen Ausbildungsbereichen vor allem dann konfrontiert, wenn die Bewertung digitalisiert und automatisiert erfolgen soll. Die Überprüfung gegenüber einer einzigen vorab angefertigten Musterlösung der Lehrkraft ist nicht zielführend, da diese nur eine einzige richtige Lösung repräsentiert, wenngleich es die einzige Möglichkeit in etablierten LMS wie z.B. Moodle ist. Es werden variable Antwortmöglichkeiten gänzlich vernachlässigt und Lernende durch ein negatives Feedback mit hoher Kontiguität der automatischen Korrektur verunsichert. Ein digitales Lehrszenario ist unter Nutzung dieser Methode nicht kompetenzorientiert und lernpsychologisch nicht förderlich [Na07]. Diesem Problem lässt sich dadurch begegnen, dass Antworten von Lernenden gegen einen möglichst großen Lösungsraum validiert werden. Somit kann zumindest *Knowledge of result* [Na07], also Feedback über die Korrektheit der Lösung gegeben werden. Dafür wurden für diesen Beitrag reguläre Ausdrücke genutzt, da diese Mengen von Zeichenketten durch syntaktische Regeln abbilden können.

2.1 Anforderungen an das Tool

Zur Entwicklung eines Prototyps wurde eine Anforderungsanalyse durchgeführt, welche die zu unterstützenden Umwandlungen und Einstellmöglichkeiten beinhaltet. Die geforderte Unterscheidung von syntaktisch notwendigen und optionalen Leerzeichen konnte technisch nicht umgesetzt werden, da je nach Kombinationsmöglichkeit der verschiedenen SQL-Strukturbausteine Leerzeichen nicht eindeutig optional sind und die technische Umsetzung nur über lokales Wissen des jeweiligen Strukturbausteins und kein

globales Wissen über die Gesamtabfrage verfügt. Daher sind alle - auch z.T. optionale - Leerzeichen immer notwendig. Optionale Alias-Namen für Spalten z.B. für die Ausgabe wurden als nicht fachlich relevant und die Umwandlung von ON-Ausdrücken in INNER JOINs zu WHERE-Ausdrücken wurde als in jedem Fall äquivalent eingestuft. Daher wurde für diese drei Anforderungen keine Einstellmöglichkeit bereitgestellt. Alle weiteren Anforderungen wurden mithilfe von Einstellungen zu- bzw. abschaltbar gestaltet, im Speziellen: Die *Rechtschreibung* der Lernenden kann in Aggregat-Funktion, Spaltenname, Funktionsname, Indexspalte, SQL-Keyword, String-Wert und Tabellenname Fehler enthalten, die ignoriert werden. Die Einstellung ermöglich hier einen Rechtschreibfehler pro genannter Struktureinheit. Variationen in der *Reihenfolge der Komponenten* in Spalten, Ausdrücken, Index-Spalten, Spalten in Group By, VALUE-Listen und Tabellennamen können ebenfalls optional berücksichtigt werden. Das Akzeptieren von *Synonymen* für Datums- und Zeitangaben, Funktionen, Aggregatfunktionen und Datentypen ist ebenfalls optional möglich. Des Weiteren werden Synonyme für sonstige SQL-Begriffe wie: Absteigend, Aufsteigend, Nicht, Und, UNIQUE, INDEX, sowie Punkt und Komma als Dezimaltrennzeichen unterstützt.

Dadurch sollen auch komplexe Strukturäquivalente erkennbar werden. Als Beispiel:

```
SELECT col1, col2 FROM tab1 INNER JOIN tab2 ON tab1.pk =
tab2.pk
SELECT      col1 AS name,col2 FROM tab1,tab2      WHERE
tab1.pk=tab2.pk
```

Dabei bleibt die Anwendung nicht auf SELECT-Statements beschränkt. Alle DDM-Statements (SELECT, INSERT, UPDATE, DELETE, sowie die DDL-Statements CREATE TABLE und DROP) sind integriert. Die Implementierung ist insofern beschränkt, dass nur die Schnittmenge aus den im schulischen Bereich viel verwendeten RDBMS-Grammatiken MySQL, MariaDB und Oracle umgesetzt wurde, was aber ohne Nachteil für die allgemeingültige Vermittlung von SQL bleibt.

2.2 Softwarearchitektur

Bei der Analyse bereits vorhandener Programmbibliotheken erwies sich der JSQLParser[5] als besonders zielführend. Dieses OpenSource-Projekt ermöglicht das Zerlegen (Parsen) und Zusammenfügen (Deparsen) von SQL-Statements auf Basis des Visitor-Patterns mithilfe einer traversierbaren Hierarchie. Damit konnte zum einen die Validierung gegenüber den RDMS-Systemen Oracle, MySQL/MariaDB sowie PostgreSQL und zum anderen die Funktionalität des Parsens von diesen syntaktisch korrekten SQL-Ausdrücken gewährleistet werden. Beim Zusammensetzen der Statements kann die benötigte Funktionalität durch Überschreiben der verwendeten Visitor-Klassen gewährleistet

[5] https://github.com/JSQLParser/JSqlParser

werden, sodass die Ausgabe eines regulären Ausdrucks möglich wird (siehe Abb. 1). Dabei besitzt jeder Statement-Typ sowie wiederkehrende Grammatik Elemente wie z.B. Ausdrücke, Group und Order By ihre eigene Visitor-Klasse.

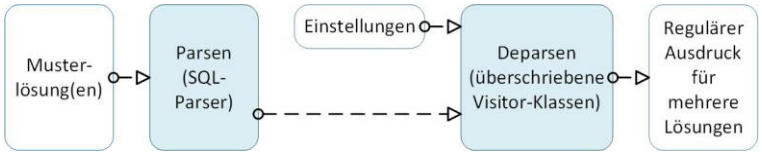

Abb. 1: Schema zur Umwandlungsstruktur

Durch die definierten Anforderungen als plattformunabhängig und die Bereitstellung des Werkzeugs als Webangebot wird das Spring-Framework als Quasi-Standard genutzt. Das System stellt ein Web-Interface zur Verfügung, über das alle Einstellungen vorgenommen, die SQL-Statements eingegeben und die regulären Ausdrücke ausgegeben werden können. Weiterhin wird eine JSON-API bereitgestellt. Diese ist in der Lage, anhand von Presets Konvertierungen vorzunehmen und das Ergebnis als JSON-Objekt zurückzuliefern.

2.3 Limitationen

Bei der Umsetzung des Tools war es nicht möglich Restriktionen bei Eingabe zu vermeiden. Das Web-Interface gibt daher den Nutzenden Hinweise in Bezug auf das für den Parser notwendige Eingabeformat. Darunter fallen Hinweise zur Angabe von:

1.) voll-qualifizierten Namen
2.) Umwandlungsrichtung für JOIN ON Ausdrücke in WHERE
3.) Constraint-Index Namen
4.) Schema-Äquivalenten (z.B. bei der Angabe von * oder Spaltennamen unter Verwendung der Count-Funktion)

Zudem sind Tabellenaliase nicht produktiv nutzbar, was im schulischen Bereich aber nahezu unbedeutend ist. Spalten-Quotings sind immer optional und damit keine Namen mit Leerzeichen möglich. Weitere Restriktionen aufgrund von Bugs sind den Issues des öffentlichen Repositorys: github.com/StEx2022-23/sql2regex zu entnehmen.

3 Evaluation

Zur Evaluation des erstellten Tools wurde es mit 46 Schülerinnen und Schülern sowie 11 Studierenden getestet. Eingeleitet wurde der Test mit der Frage nach Klassenstufe bzw. Alter, vorhandenem Vorwissen in SQL allgemein und e-Assessment Szenarien in SQL. Dabei nahmen insgesamt 46 Schülerinnen und Schüler aus zwei Gymnasien sowie davon 11 Studierende der TU-Dresden teil. Zu etwa 50% wurden Vorerfahrungen im Umgang mit digitalen Übungs- und/oder Prüfungsszenarien zum Thema SQL angegeben. Diese

bezogen sich auf die Tools InstaHub[6], SQLIsland[7] und Opal. Die Aufgabenverteilung der Statements umfasste 5x SELECT, 2x INSERT INTO, 2x UPDATE, 2x DELETE, 1x CREATE DATABASE, 1x CREATE TABLE, 1x DROP TABLE.

Alle Antworten zu Statements gleicher Art wurden dabei für die Auswertung zusammengefasst, da sie auf gleichen DeParser-Klassen beruhen. Die Übergewichtung in Select-Statements beruht auf der zu erwartenden Gewichtung im Schulunterricht. Dabei wird für die Wirksamkeit der Überprüfung durch die generierten regulären Ausdrücke die Handlungsalternativen für einen Test ohne zusätzliche technische Kenntnisse und Ressourcen zwischen *Erkennung ohne RegEx*, *Erkennung ohne RegEx mit Case-Sensitivity* und *Erkennung mittels RegEx* unterschieden. Nach der Auswertung von insgesamt 542 Antworten ergibt sich im Maximum eine signifikante Verbesserung der Validierungsleistung auf 90% anstatt 10% mit herkömmlichen Kontrollmethoden. Ein Verzicht auf die Prüfung von Groß- und Kleinschreibung bei einem Vergleich der Zeichenketten führt zu einer minimal besseren Erkennung richtiger Antworten, allerdings nicht vergleichbar mit den Ergebnissen der Validierung über die Nutzung von regulären Ausdrücken (vgl. Abb. 2).

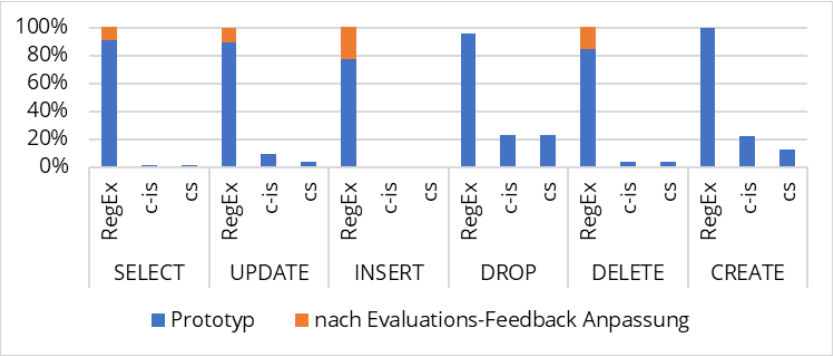

Abb. 2: richtig validierte SQL-Statements mithilfe regulärer Ausdrücke (RegEx), zeichengetreue Kontrolle case-sensitive (c-is) und case-sensitive (cs)

Zusätzlich wurden in der Stichprobe keine falsch-positiven Lösungen detektiert, was vor allem für den Einsatz in Prüfungsszenarien von Bedeutung ist. Allerdings wurden einige falsch-negative Lösungen detektiert, so dass 1.) das Zulassen von Leerzeichen vor und hinter Statements, 2.) die Ergänzung fehlender Zeit- und Datumsformate und 3.) Äquivalenzen aufgrund des Datenbank-Schemas hinzugefügt wurden. Damit konnte die Validierungsleistung für Select, Insert, Update, Delete-Statements im Rahmen der Stichprobe auf 100% gesteigert werden (vgl. Abb. 2).

[6] https://instahub.org/
[7] https://sql-island.informatik.uni-kl.de/

4 Diskussion

Durch den Ansatz über reguläre Ausdrücke konnte eine deutliche Steigerung korrekten Feedbacks gegenüber der zeichengetreuen Kontrolle mit einer Musterlösung erzielt werden. Die bereits guten Ergebnisse des Prototyps konnten nach den wenigen durch den Usertest aufgedeckten Schwachstellen weiter verbessert werden. Das Tool bietet eine größere Flexibilität, da es keine Datenbankimplementierung benötigt sowie den Vorteil der inneren Differenzierung - bspw. über die Einstellungsmöglichkeiten zur Rechtschreibung - gegenüber Ansätzen der Validierung über die Anzahl der Tupel und Attribute, wie es z.B. bei SQL-Island der Fall ist. Für eine vollumfängliche Anwendung im universitären Bereich müsste jedoch als Beispiel die zusätzliche Implementierung der Umwandlung für *Alter Table* bedacht werden. Trotz der theoretischen Einsatzfähigkeit in der Schule steht der praktischen Anwendungen in diesem Bereich die noch fehlende Integration regulärer Ausdrücke in alle bestehende Learning Management Systeme entgegen. Für diese Herausforderung könnte nach dem mit diesem Beitrag erfolgten Proof of Concept die bereits vorhandene JSON-API um die Validierung von Statements erweitert werden. Mithilfe dieser Brückentechnologie könnte der Nutzendenkreis auf alle Umgebungen erweitert werden, welche eigene JavaScript-Snippets erlauben.

Literaturverzeichnis

[Na07] Narciss, S.: Feedback Strategies for Interactive Learning Tasks. In (Driscoll, M. P.; Jonassen, D. H.; Harris, P. Hrsg.): Handbook of Research for Educational Communications and Technology. A Project of the Association for Educational Communications and Technology. Taylor & Francis, Hoboken, S. 125–143, 2007.

[Na22] S. Nayak, R. Agarwal and S. K. Khatri, "Review of Automated Assessment Tools for grading student SQL queries," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-4, doi: 10.1109/ICCCI54379.2022.9740799.