



Toolunterstützung für den Übungsbetrieb in der Datenbanklehre: Erfahrungen mit der Software *Praktomat*

Christina Ehrlinger¹ · Thomas Fritsch² · Michael Fruth³  · Franz Lehner² · Stefanie Scherzinger³

Eingegangen: 4. November 2020 / Angenommen: 11. Februar 2021 / Online publiziert: 3. März 2021
© Der/die Autor(en) 2021

Zusammenfassung

In diesem Artikel berichten wir über den erstmaligen Einsatz der Software *Praktomat* im Rahmen einer einführenden Datenbankvorlesung an der Universität Passau. Unser Ziel war es, den ca. 300 Studierenden aus Informatik und Wirtschaftsinformatik während des „Corona-Semesters“ die Möglichkeit zu bieten, Anfragen in relationaler Algebra und SQL im Selbststudium zu üben und dabei automatisierte Rückmeldungen zu bekommen. Insbesondere wollten wir kein neues E-Learning-System von Grund auf implementieren, sondern auf einem bestehenden, quelloffenen Werkzeug aufsetzen. Wir beschreiben in diesem Artikel die notwendigen Anpassungen der *Praktomat*-Software, ursprünglich für die grundständige Programmierausbildung konzipiert. Wir evaluieren den Zufriedenheitserfolg unserer Studierenden und berichten weiter anekdotisch über die Erfahrungen der Studierenden und Lehrenden.

Schlüsselwörter Datenbanklehre · Praktomat

1 Einleitung

Die Bedingungen für den Einsatz von E-Learning an Hochschulen haben sich in den letzten Jahren stark gewandelt [1]. Die Situation ist zwar weiterhin durch Pilotanwendungen und explorative Projekte bestimmt, doch zeichnet sich ein

Trend zum nachhaltigen Einsatz mit institutioneller Verankerung ab.

In diesem Artikel berichten wir über solch ein exploratives Projekt im Rahmen einer Bachelor-Grundlagenvorlesung mit begleitender Übung: Im Sommersemester 2020 besuchten ca. 300 Studierende der Universität Passau, überwiegend aus den Informatik- und Wirtschaftsinformatikstudiengängen, die einführende Datenbankvorlesung. Coronabedingt wurde die Veranstaltung erstmals komplett virtuell durchgeführt.

Für das Üben der relationalen Algebra und der Anfragesprache SQL benötigten wir kurzfristig eine geeignete E-Learning Software. Aufgrund des Zeitdrucks (de-facto eine Vorlaufzeit von zwei Wochen) war eine Neuentwicklung ausgeschlossen. Daher entschieden wir uns, ein bestehendes System anzupassen.

Konkret setzten wir auf die quelloffene Software *Praktomat*. Diese Wahl war naheliegend, denn *Praktomat* wurde ursprünglich an der Universität Passau entwickelt [2], und wird seit über zwei Jahrzehnten in der grundständigen Programmierausbildung an der Fakultät für Informatik und Mathematik eingesetzt: Die Studierenden reichen ihre Programmierlösungen in einem web-basierten Abgabesystem ein und erhalten unmittelbar die Ergebnisse einer automatischen Überprüfung.

Christina Ehrlinger
christina.ehrlinger@uni-passau.de

Thomas Fritsch
thomas.fritsch@uni-passau.de

✉ Michael Fruth
michael.fruth@uni-passau.de

Franz Lehner
franz.lehner@uni-passau.de

Stefanie Scherzinger
stefanie.scherzinger@uni-passau.de

¹ Lehrstuhl für Informatik mit Schwerpunkt Informationsmanagement, Universität Passau, Passau, Deutschland

² Lehrstuhl für Wirtschaftsinformatik mit Schwerpunkt Informations- und IT-Service-Management, Universität Passau, Passau, Deutschland

³ Lehrstuhl für Informatik mit Schwerpunkt Skalierbare Datenbanksysteme, Universität Passau, Passau, Deutschland

Mittlerweile wird *Praktomat* am Karlsruher Institut für Technologie weiterentwickelt¹ und ist deutschlandweit an Hochschulen im Einsatz. *Praktomat* ist ein Vertreter einer ganzen Familie solcher E-Learning-Systeme und wir verweisen auf [3–5] für einen Überblick von Evaluationstools für die Programmierausbildung.

Im Folgenden berichten wir von unseren Erfahrungen mit *Praktomat* in der Datenbanklehre, in Hinblick auf eine Verstetigung über das erste „Corona-Semester“ hinaus. Wir setzen dabei folgende Schwerpunkte:

- Wir geben einen Überblick über die technischen Anpassungen von *Praktomat*. Letztendlich waren diese mit vertretbarem Aufwand machbar, insbesondere war es nicht nötig, in den Programmcode von *Praktomat* einzugreifen. Wir stellen unsere Anleitungen und Skripte auf GitHub zur Verfügung.
- Wir evaluieren den Einsatz von *Praktomat* anhand eines wissenschaftlich etablierten Fragebogens und messen den Zufriedenheitserfolg der Studierenden.
- Wir analysieren *Praktomat* Nutzungsdaten und geben Rückmeldungen aus der Lehrevaluation wieder.
- Wir diskutieren mögliche Erweiterungen und Fragestellungen als Teil des Ausblicks unserer Arbeit.

Struktur. Unser Bericht ist wie folgt strukturiert. In Kap. 2 geben wir einen kurzen Überblick über bestehende Software-Werkzeuge. In Kap. 3 skizzieren wir unsere Erweiterungen von *Praktomat*. Kap. 4 stellt die Ziele und Ergebnisse unserer Evaluation vor. Im Folgekapitel bereiten wir weitere Ergebnisse auf, basierend auf einer Analyse der *Praktomat* Nutzungsdaten und Rückmeldungen aus der Lehrevaluation. In Kap. 6 diskutieren wir zukünftige mögliche Erweiterungen.

2 Software-Werkzeuge für das Üben von relationaler Algebra und SQL

Wir geben einen kurzen Überblick über gängige Software-Werkzeuge für das Üben von relationalen Anfragesprachen. Der von uns gewählte Ansatz unterscheidet sich von existierenden E-Learning-Systemen darin, dass wir ein bestehendes System, den *Praktomat*, für das Überprüfen von Datenbankabfragen erweitern, anstatt ein neues System von Grund auf zu implementieren. Zudem bieten wir Unterstützung sowohl für relationale Algebra als auch für SQL, während bestehende Systeme sich meist nur einer Sprache widmen.

Relationale Algebra. Mittlerweile gibt es diverse Software-Werkzeuge für das Formulieren von Anfragen in relationaler Algebra, z. B. [6–8]. Wir haben bereits in frühe-

ren Semestern die web-basierten Systeme *RelaX* [6] und *IRA* [7] erfolgreich in der Datenbanklehre eingesetzt, sie stellen die einzelnen Auswertungsschritte anschaulich und nachvollziehbar dar. Bei der Integration mit *Praktomat* hingegen kam es uns nicht auf eine interaktive Visualisierung an, sondern auf die einfache Integrierbarkeit mit Shell-Skripten. Wir haben uns aus diesem Grund für *radb* [8] entschieden, da wir bereits in einer früheren Lehrveranstaltung gute Erfahrungen mit der Einbettung von *radb* in größere Software-Architekturen gemacht haben [9, 10]. *radb* ist sowohl in Python also auch in Java implementiert und delegiert intern die Ausführung von Anfragen an eine SQLite Datenbank.

Die Syntax von *radb* erinnert an die Syntax von \LaTeX , wie folgendes Beispiel zeigt.

Beispiel 1 Die Anfrage $\pi_{\text{pizza}}((\sigma_{\text{age}=20}\text{Person}) \bowtie \text{Eats})$ („Welche Pizzen essen die 20-jährigen Personen?“) wird in *radb* Syntax wie folgt formuliert.

```
\project_{pizza}{
  (\select_{age=20} Person) \join Eats}
```

Zusammenfassend ergibt sich daher eine gute Eignung von *radb* für unser Vorhaben.

Structured Query Language (SQL). Für das Üben von SQL-Anfragen stehen diverse Software-Werkzeuge zur Verfügung, zum Teil mit langjähriger Tradition (entsprechend gibt es frühere Artikel im Datenbankspektrum, die Werkzeuge zum Üben von SQL beschreiben, z. B. [11]). Für einen aktuellen und umfassenden Überblick verweisen wir auf den Artikel zum *SQLValidator* [12] in diesem Themenheft.

Generieren von erklärenden Beispielen. Es sind eine Reihe interessanter Forschungsarbeiten zum Generieren von Gegenbeispielen beim Formulieren von Datenbankabfragen auf Basis von *radb* entstanden [13, 14]. Diese Werkzeuge sind zum aktuellen Zeitpunkt nur als online Demos², jedoch nicht als quelloffene Software verfügbar. Die Idee, Anfragen zu erklären bzw. zu debuggen, ist verwandt mit den Fragestellungen nach *Provenance* bei der Auswertung von SQL-Anfragen, z. B. [15, 16]. Auch hier ist uns keine frei verfügbare Software-Bibliothek bekannt, welche ad-hoc in eine bestehende Software wie *Praktomat* integrierbar wäre.

Das Generieren von Hilfestellungen bei der Anfrageformulierung wirft im Zeitalter der Massive Open Online Course (MOOC)s praktisch relevante und herausfordernde theoretische Fragestellungen auf – hinter denen zudem auch ein kommerzielles Interesse steckt. Während wir ge-

¹ <https://github.com/KITPraktomatTeam/Praktomat>.

² Online Demo unter <https://dukedb-hnrq.github.io/>, aufgerufen am 10.12.2020.

genwärtig diese Techniken noch nicht einsetzen, skizzieren wir entsprechende Pläne als Teil des Ausblicks.

3 Die Software *Praktomat*

Wir beschreiben kurz die *Praktomat*-Software, bevor wir auf die Anpassungen für die Datenbanklehre eingehen.

3.1 Systemüberblick und Einsatzform

Praktomat wurde ursprünglich zur Unterstützung der Programmierausbildung entwickelt [2, 17, 18]. Wir geben im Folgenden nur einen Überblick über das System, und verweisen auf [19] für die technischen Details.

Das quelloffene System ist als Webdienst realisiert. Genauer ist *Praktomat* eine Django-Anwendung, die in Verbindung zu einer Datenbank steht und üblicherweise auf einem Linux-Server betrieben wird.³ Pro Lehrveranstaltung ist eine eigene Instanz des Dienstes anzulegen, es ist jedoch problemlos möglich, mehrere Instanzen auf einem Server zu betreiben. *Praktomat* ermöglicht das Ausführen von studentischem Code innerhalb einer Sandbox, sowie eines Docker Containers. Die Aufgaben selbst können über eine Webchnittstelle angelegt werden.

Der Ablauf aus studentischer Sicht ist wie folgt: Die Studierenden reichen per Upload ihre Programmierlösungen zur Überprüfung ein. Der Programmcode (z. B. Java) wird auf dem *Praktomat*-Server kompiliert und Funktionstests (z. B. JUnit) unterzogen. Die Studierenden bekommen daraufhin Rückmeldung, ob ihr Programmcode die Tests besteht.

3.2 Anpassungen für die Datenbanklehre

Praktomat sieht „out-of-the-box“ keine Überprüfung von Datenbankabfragen vor. Im Folgenden skizzieren wir unsere Anpassungen von *Praktomat* für das Überprüfen von Abfragen in relationaler Algebra und SQL. Eine technisch detailliertere Beschreibung, sowie konkrete Skriptvorlagen, stellen wir auf GitHub zur Verfügung.⁴

Relationale Algebra. Wir verwenden `radb` für das Üben von relationaler Algebra und haben das `radb` Python-Modul in das *Praktomat*-System integriert. `radb` übersetzt relationale Algebra nach SQL und führt die SQL-Anfragen auf einer eingebetteten SQLite Datenbank aus.

Praktomat vergleicht anschließend das Anfrageergebnis mit einer Musterlösung. Dazu nutzen wir den *Script-Che-*

cker von *Praktomat*, ein eingebauter Überprüfungsschritt, der es uns erlaubt, ein beliebiges Shell-Skript aufzurufen. Unser Shell-Skript delegiert die Auswertung der studentische Anfrage an das `radb` Modul. Schlägt die Ausführung fehl (z. B. wegen eines Syntaxfehlers), wird die Fehlermeldung zurückgeliefert. Andernfalls wird das von `radb` produzierte Anfrageergebnis in eine Datei umgeleitet. Diese Datei wird mit einer Musterlösung verglichen. Dass in der Auswertung der relationalen Algebra die Reihenfolge von Tupeln oder Attributen irrelevant ist, wird berücksichtigt. Falls die studentische Anfrage nicht das erwartete Ergebnis liefert, wird eine entsprechende Rückmeldung generiert. Sollte z. B. ein Tupel fehlen, wird „Mindestens ein Tupel fehlt“ zurückgemeldet. Wenn die Muster- und die studentische Lösung (bis auf die Reihenfolge von Tupeln) übereinstimmen, gilt die Aufgabe als erfolgreich gelöst.

SQL. `radb` delegiert Anfragen an SQLite, doch für das Üben von SQL-Anfragen ist SQLite als ausführendes Datenbankmanagementsystem unserer Erfahrung nach weniger geeignet. Zwar implementiert SQLite größtenteils die PostgreSQL Syntax, doch werden die Studierenden an SQLite-spezifische Eigenheiten gewöhnt. So gibt es nur den linken äußeren Join, nicht aber den rechten oder vollen äußeren Join.

Aus didaktischer Sicht besonders problematisch ist, dass bei GROUP-BY-Anfragen in der SELECT-Klausel auch Attribute erlaubt sind, die weder aggregierte Werte noch Gruppierungsattribute darstellen⁵, wie folgendes Beispiel illustriert.

Beispiel 2 Anfragen nach folgendem Muster werden in SQLite ohne Fehlermeldung ausgewertet:

```
SELECT name, count(*) AS personcount
FROM Person
GROUP BY age;
```

Gleichzeitig sind aber gerade solche Gruppierungsanfragen für Studierende oft eine Herausforderung, wie systematische Analysen zeigen [20, 21]. Da wir eine Behandlung als Fehlerfall hier für sinnvoll erachten, haben wir uns für PostgreSQL und gegen SQLite entschieden.

Da PostgreSQL serverbasiert ist, ist die Einbindung in das *Praktomat*-System aufwändiger: Aus Sicherheitsbedenken wird nicht das *Praktomat*-interne Datenbankmanagementsystem mitbenutzt, sondern eine eigene Instanz innerhalb eines Docker-Containers verwendet.

Der weitere Ablauf ähnelt dem der Überprüfung von Abfragen in relationalen Algebra, den Kern bildet wieder ein

³ Wir verwenden eine virtuelle Maschine mit 2 CPUs und 4 GB RAM, mit Ubuntu 18.04 LTS und PostgreSQL Version 11.7 als Datenbankmanagementsystem.

⁴ <https://github.com/sdbs-uni-p/praktomat-for-radb-and-psql>.

⁵ https://sqlite.org/quirks.html#aggregate_queries_can_contain_non_aggregate_result_columns_that_are_not_in_the_group_by_clause, aufgerufen am 10.12.2020.

Vergleich mit einer Musterlösung. Unser Shell-Skript verwaltet den Zugriff auf die PostgreSQL Instanz, die in einer Docker-Umgebung gestartet wird. Pro Aufgabe können die benötigten Lese- bzw. Schreibrechte definiert werden. Bei reinem Lesezugriff kann eine gemeinsame Instanz genutzt werden, wohingegen beim Schreibzugriff jede Abgabe auf einer temporär angelegten Instanz ausgeführt wird. Dadurch wird sichergestellt, dass Studierende die Infrastruktur nicht manipulieren können.

Der Vergleich mit der Musterlösung berücksichtigt die Ordnung der Ergebnistupel, sofern dies für die zu überprüfende Anfrage konfiguriert wurde.

Vorgängerprojekte. Uns ist nur ein früheres Projekt bekannt, in dem *Praktomat* für die Auswertung von SQL-Anfragen angepasst wurde [22]. Dieses System wurde jedoch laut den Autoren an der Ostfalia Hochschule für angewandte Wissenschaften bisher noch nicht produktiv in der Lehre eingesetzt. Die Vorgehensweise bei diesem Vorgängerprojekt ist ähnlich zu unserer, sofern wir das auf Grundlage von [22] beurteilen können: Die Ausführung von SQL-Anfragen wird ebenfalls an ein relationales Datenbankmanagementsystem (Oracle) delegiert, im Anschluss werden die Anfrageergebnisse mit einer Musterlösung verglichen.

3.3 Aufgabenstellungen

Praktomat wurde im Sommersemester 2020 in zwei Aufgabenrunden eingesetzt. In der ersten Runde waren 16 Anfragen in relationaler Algebra zu formulieren, in der zweiten Runde weitere 26 Anfragen in SQL. Die Anfragen bauen auf Szenarien von Jennifer Widoms MOOC auf (wie etwa das „pizza“ Szenario aus den Beispielen 1 und 2) [23].

Beide Anfragesprachen konnten außerhalb des *Praktomat*-Systems geübt werden: So konnten die Studierenden frei mit der Anfragesprache experimentieren. Für Anfragen in *radb* unterstützten wir die Studierenden darin, das Python Modul auf ihrem eigenen Rechner zu installieren. (Künftig wollen wir zusätzlich eine Webschnittstelle für *radb* anbieten, weil Studierende aus anderen Fakultäten, die unsere Vorlesung besuchten, mit der Benutzung der Kommandozeile und der Installation von Python mitunter Schwierigkeiten hatten.)

Für das Üben von Anfragen in SQL verweisen wir auf die Webseite www.db-fiddle.com, die Zugang zu einem PostgreSQL System über den Browser ermöglicht.

3.4 Sicherheit und Datenschutz

Praktomat verfügt über eine eigene Benutzerverwaltung und verarbeitet daher personenbezogene Daten: Es werden Namen, Matrikelnummern, E-Mail-Adressen sowie die studentischen Abgaben in der *Praktomat*-eigenen Datenbank erfasst. Mit Hilfe eines regulären Ausdrucks kann si-

chergestellt werden, dass sich nur Studierende mit einer E-Mail-Adresse der Universität Passau registrieren. Durch einen Bestätigungslink, enthalten in einer Registrierungs-mail, wird die Echtheit der E-Mail Adresse überprüft. Zudem ist unser *Praktomat*-Server nur innerhalb des Universitätsnetzes erreichbar.

Alle Maßnahmen zur Speicherung und Verarbeitung von personenbezogenen Daten wurden durch die Datenschutzbeauftragten der Universität Passau freigegeben.

Zuletzt müssen die Nutzungsdaten langzeitarchiviert werden, da die studentischen Lösungen Teil der Prüfungsleistung sind. Dazu wird die virtuelle Maschine zum einen durch das Rechenzentrum der Universität archiviert und zum anderen auf einem Datenträger gesichert und gemeinsam mit den Klausuren archiviert.

4 Evaluation des Zufriedenheitserfolgs

Eine wesentliche Rolle für die nachhaltige Nutzung von E-Learning-Systemen wie *Praktomat* spielen positive Nutzungserfahrungen der Studierenden. Die Evaluation und Bewertung von E-Learning bzw. des Einsatzes von E-Learning-Systemen ist allerdings komplex und mit methodischen Herausforderungen verbunden. Bei einem traditionellen Vorgehen wird meist versucht, direkt nach dem Nutzen zu fragen, ohne vorab das Ziel der Untersuchung klar zu spezifizieren. Dies ist jedoch wesentlich, da mit dem Nutzen recht unterschiedliche Ausprägungen wie Benutzerzufriedenheit, Qualität des Systems, Lernerfolg, Lernaufwand u. a. m. verbunden sein können. Die Ergebnisse können so zur Verbesserung des Systems, aber auch der Inhalte und Abläufe herangezogen werden. Dies lässt einen Wandel von reiner Design-Orientierung (Fokus auf die technische Funktionalität) hin zu einer Didaktik-Orientierung erkennen.

Abb. 1 visualisiert diese Zusammenhänge: Die Evaluationsergebnisse sollen als Basis für eine gezielte Verbesserung der Lernsituation dienen. Wir unterscheiden technische, inhaltliche und didaktische Verbesserungen. Alle drei Aspekte sind wichtig, um den Bedürfnissen der Teilnehmenden gerecht zu werden.

Kirkpatrick [24] schlägt zur Orientierung ein 4-Ebenen-Modell zur Einordnung vor (siehe Abb. 2). Es wird davon ausgegangen, dass jede Evaluationsstufe auf den Ergebnissen der darunterliegenden Ebene aufbaut.

In einem vollständigen Evaluationsprozess werden alle vier Ebenen nacheinander durchlaufen:

1. Die Basis bildet der **Zufriedenheitserfolg** aus Sicht der teilnehmenden Studierenden.
2. Die Ebene darüber ist der eigentliche **Lernerfolg**, dieser sollte sich im Klausurerfolg widerspiegeln.

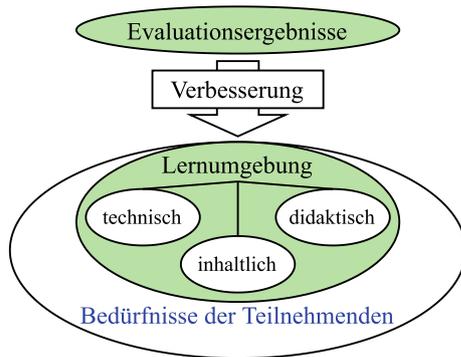


Abb. 1 Unterscheidung von Zielen bei der Evaluation von E-Learning-Maßnahmen

3. Im **Transfererfolg** wird betrachtet, was vom Gelernten über die Klausur hinaus beherrscht wird.
4. Die oberste Ebene erfasst die Ergebnisse im Sinne der Erreichung der institutionellen Ziele. Im Unternehmenskontext kann dies der Beitrag zum **Geschäftserfolg** sein, im Kontext einer Universität der Beitrag zu den im Rahmen der Strategie definierten E-Learning-Zielen.

Die Komplexität und der Aufwand einer Messung nehmen mit jeder Ebene von unten nach oben zu.

Eine wichtige Frage ist in diesem Zusammenhang, wofür die Untersuchungsergebnisse verwendet werden sollen. Im vorliegenden Fall geht es uns primär um die Zufriedenheit der Teilnehmenden, die Nutzungsintensität (*Wie gehen die Lernenden mit dem Lehrangebot um?*) und den Beitrag des *Praktomat*-Systems zum Lernerfolg. Aus dem Ziel (*Was misst man?*) leitet sich dann die Methode (*Wie misst man?*) ab. Sinnvoll ist somit ein methodisch unterstütztes Vorgehen, bei dem auf Basis vorab festgelegter Ziele eine situationsspezifische Anpassung eines Messmodells vorgenommen wird.

Unsere Untersuchung umfasst die Ebenen 1 und 2 im Modell von Kirkpatrick. Auf der ersten Ebene geht es insbesondere darum, wie die Lernenden auf den Ablauf reagieren, d.h. wie zufrieden sind die Lernenden mit *Praktomat* und seinen spezifischen Elementen? Es geht also um Akzeptanz, Zufriedenheit und empfundenen Nutzen. Auf der zweiten Ebene stellt sich die Frage, ob sich die Kenntnisse und Fähigkeiten der Lernenden verbessert haben. Es geht somit um subjektiven und objektiven Lernerfolg.

In Verbindung mit E-Learning-Maßnahmen gilt die System-Qualität allgemein als der wichtigste Faktor in Bezug auf die Zufriedenheit, die Nutzungsintensität und den Nutzen und zwar sowohl für einzelne Nutzerinnen oder Nutzer, als auch aus der Perspektive der anbietenden Organisation.

Tatsächlich existieren nur wenige Modelle für die Evaluation von E-Learning-Systemen (vgl. [25]). Für unsere Befragung verwenden wir daher ein Messmodell, das sich

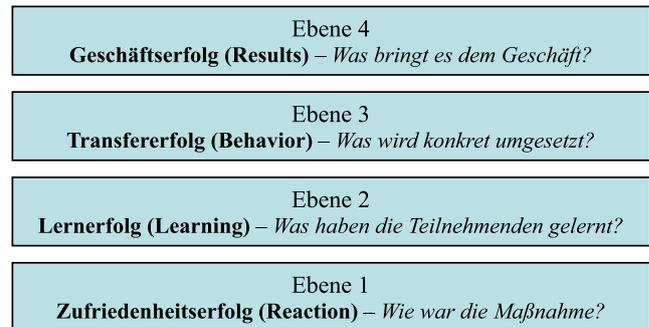


Abb. 2 4-Ebenen-Modell von Kirkpatrick [24]

bereits im Hochschulumfeld bewährt hat und speziell für diesen Zweck entwickelt worden ist [25]. Es baut auf dem in der Wirtschaftsinformatik verbreiteten Modell für „IS-Success“ nach Delone und McLean auf (siehe z. B. [26, 27]) und ist für die Verwendung in Verbindung mit E-Learning entsprechend adaptiert [25]. Insbesondere weist dieses Modell gute psychometrische Eigenschaften auf.

Unser Fragebogen besteht vorwiegend aus Zustimmungsfragen und ist online verfügbar.⁶

Auswertung. An der Umfrage nahmen 43 Studierende teil. Nachdem die Vorlesung das erste Mal mit *Praktomat*-Unterstützung angeboten wurde, können wir die Ergebnisse nicht mit Vorjahresdaten vergleichen.

Im Folgenden fassen wir die Ergebnisse zusammen. In Abb. 3 visualisieren wir zwei Teilfragen. Die größtenteils im 2. Semester eingeschriebenen Studierenden geben eine hohe Zufriedenheit (Mean = 3,79, siehe Abb. 3a) mit *Praktomat* an und würden diesen weiterempfehlen (Mean = 3,93) bzw. diesen auch gerne in anderen Kursen nutzen (Mean = 3,86). In puncto Zuverlässigkeit (Mean = 4,05), Stabilität (Mean = 4,37), Verfügbarkeit (Mean = 4,51) und Einfachheit der Bedienung (Mean = 3,98) erzielt *Praktomat* ebenfalls eine gute Wertung.

Im Mittel erfüllt *Praktomat* mit einem Wert von 3,76 die Erwartungen der Studierenden und verbessert nach Einschätzung der Studierenden Nachhaltigkeit (Mean = 4,15, siehe Abb. 3b), Effektivität (Mean = 4,22) sowie Effizienz (Mean = 3,93) beim Lernen. Das Ziel positiver Nutzungserfahrungen mit *Praktomat* kann damit auf jeden Fall als erreicht angesehen werden.

⁶ Fragebogen online verfügbar unter: <https://github.com/sdbs-uni-p/praktomat-questionnaire-2020>.

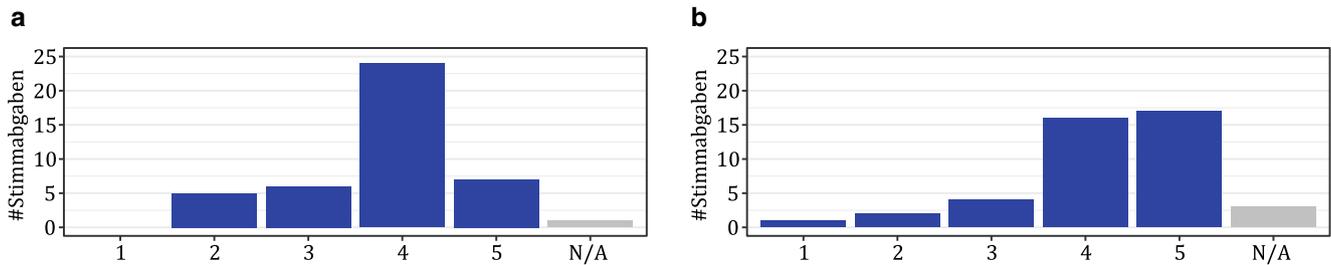


Abb. 3 Ausgewählte Fragen aus dem Fragebogen, Spektrum der Antworten jeweils von „1: Stimme gar nicht zu“ bis „5: Stimme uneingeschränkt zu“, sowie keine Angabe (N/A). **a** „Sie sind mit dem *Praktomat*“ zufrieden, **b** „*Praktomat*“ verbessert die Nachhaltigkeit beim Lernen

5 Weitere Ergebnisse

Im Folgenden stellen wir weitere Ergebnisse vor, basierend auf der Auswertung der *Praktomat*-Nutzungsdaten sowie Rückmeldungen aus der Lehrevaluation.

5.1 Analyse der *Praktomat* Nutzungsdaten

Es nahmen insgesamt 121 Studierende von der Möglichkeit Gebrauch, ihre Anfragen durch *Praktomat* prüfen zu lassen, um damit Bonuspunkte für die Klausur zu erwerben. Diese Studierenden haben mindestens einen Lösungsversuch eingereicht.

Für die 42 zu lösenden Aufgaben wurden insgesamt ca. 7.800 Lösungsversuche eingereicht. Die 75% der Studierenden, die alle Aufgaben erfolgreich bearbeiten konnten, brauchten im Schnitt nur knapp zwei Versuche pro Aufgabe. Trotz dieser geringen Anzahl an Versuchen gibt es kaum konkrete Hinweise auf Täuschungsversuche: Unter allen teilnehmenden Studierenden konnten wir nur drei Paare identifizieren, die durchgängig dieselben Lösungen hochgeladen haben. Unsere Vermutung ist, dass die meisten Studierenden an den Aufgaben außerhalb des *Praktomat*-Systems gearbeitet haben (wie in Kap. 3.3 beschrieben) und nur aussichtsreiche Lösungen zur Überprüfung eingereicht haben. Dadurch können wir auch die tatsächliche Bearbeitungszeit einzelner Aufgaben nicht erfassen.

Die Aufgaben haben unterschiedliche Schwierigkeitsgrade, und wir visualisieren die Anzahl der Lösungsversuche für die 16 *radb* Aufgaben, differenziert nach Studierenden und Aufgaben, in Abb. 4. Der Farbton gibt dabei die Anzahl der Versuche bis zur ersten Lösung an. Die Skala reicht von null Versuchen (heller Farbton) bis acht (oder mehr) Versuchen (dunkler Farbton). Abgaben, die zu keiner akzeptierten Lösung geführt haben, werden als null Versuche gewertet. Das markierende Dreieck trennt die Studierenden, die alle Aufgaben lösen konnten (links) von denen, die mindestens eine Aufgabe nicht lösen konnten (rechts).

86 Studierende konnten alle 16 *radb*-Aufgaben lösen, 28 scheiterten, darunter 8 eher knapp, sie hatten mehr als 80% der Aufgaben gelöst. In Abb. 4 ist zudem die Tendenz

erkennbar, dass der Schwierigkeitsgrad der Aufgaben mit aufsteigender Nummerierung zunimmt.

Die Verteilung der 26 SQL-Aufgaben ist ähnlich zu *radb*, es hatten sich 99 Studierende daran versucht, etwa 75% der teilnehmenden Studierenden konnten alle Aufgaben lösen. Nur 3 der gescheiterten Studierenden hatten mehr als 80% der Aufgaben gelöst.

Insgesamt konnten 68 Studierende alle Aufgaben zu *radb* und SQL lösen.

5.2 Lehrevaluation

Feedback aus der Lehrevaluation zum Einsatz von *Praktomat* können wir nur anekdotisch wiedergeben. Es gab Lob, aber auch Kritik. Zum einen wurde bemängelt, dass der Einsatz von *Praktomat* zur Medienvielfalt beigetragen hätte, weil ein weiteres Tool bedient werden musste. Das war im Corona-Semester, und der allgemeinen „Zoom-Fatigue“, sicherlich ein Aspekt.

Eine weitere Rückmeldung betrifft das eingeschränkte Feedback durch *Praktomat*, so wurde moniert: „Bei einem falschen Ergebnis gab es auch keine Begründung, wo der Fehler denn gelegen ist und wie man diesen möglicherweise lokalisieren kann.“ Dass es sich bei der Generierung von hilfreichen Erklärungen und Beispielen noch um ein aktives Forschungsgebiet handelt (vgl. Kap. 2), ist für Studierende im zweiten Semester wohl wenig tröstlich.

6 Diskussion und Ausblick

Abschließend diskutieren wir unsere Beobachtungen und mögliche weitere Schritte.

Didaktik. Wir haben unsere Aufgabenstellungen aus unserer z.T. mehrjährigen Lehrerfahrung heraus formuliert. Allerdings gibt es konkrete, systematische Ansätze, die Lernzieltaxonomie von Bloom [28] auf die Generierung von Aufgabenstellungen für das Erlernen und Prüfen von SQL-Anfragen anzuwenden (z. B. [29, 30]). Wir halten eine Überarbeitung unserer Aufgabensammlung unter diesen Gesichtspunkten für vielversprechend.

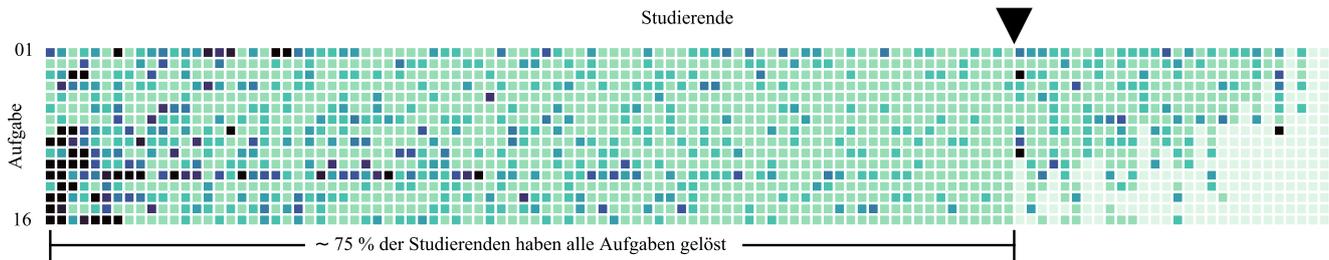


Abb. 4 Anzahl der Abgaben eines Studierenden für alle 16 *radb*-Aufgaben. Ein heller Farbton gibt keine Abgaben an, oder Abgaben, die zu keiner Lösung geführt haben. Der dunkelste Farbton repräsentiert acht oder mehr Abgaben. Die Markierung teilt die Studierenden in zwei Gruppen auf: Studierende links der Markierung konnten alle Aufgaben lösen, Studierende rechts der Markierung haben nicht alle Aufgaben erfolgreich gelöst

Plagiarismus. Im Fall der beschriebenen Lehrveranstaltung konnten die Studierenden durch erfolgreiche Abgabe der *Praktomat*-Aufgaben Bonuspunkte erwerben, die bei der Korrektur der Klausuren berücksichtigt wurden. Mitunter beobachteten wir allerdings bei der Berechnung der Endnote Diskrepanzen zwischen dem Erfolg bei den *Praktomat*-Aufgaben und dem Klausurerfolg. Das wirft die Frage auf, ob trotz des allgemeinen Zufriedenheitserfolgs der Lernerfolg letztlich ausbleibt, oder ob diese Studierenden für die *Praktomat*-Aufgaben fremde Lösungen kopiert haben.

Plagiarismus im Informatikstudium ist ein bekanntes Problem [31] und für die automatische Bewertung von Programmieraufgaben sind diverse Plagiats-Checker verfügbar, wie etwa JPlag [32]. Auch für SQL-Anfragen wurden Plagiats-Checker vorgeschlagen, z. B. [33, 34]. Allerdings ist die Aussagekraft gerade bei Aufgaben auf Anfängerniveau beschränkt: Mitunter umfasst der Lösungsraum, die Anzahl der unterschiedlichen Formulierungen einer SQL-Anfrage (bis auf Groß-Kleinschreibung) nur wenige Zeichenketten, so dass ein Plagiats-Checker zu viele Falsch-Positive generieren würde. Erste Studien in diese Richtung bestätigen diese Hypothese [35].

Ein weiteres, aus der Programmierausbildung bekanntes Problem ist das Umgehen der automatisierten Prüfung, indem die Lösung konstruiert wird [36]. Übertragen auf Abfragesprachen würden dann die erwarteten Tupel über ihren Schlüssel direkt selektiert. Allerdings konnten wir diese Art der Umgehungsversuche in unserer Analyse nicht feststellen.

Code Reviews. Eine weitere Fragestellung ist, inwiefern Code Reviews dabei helfen, Datenbankabfragen formulieren zu lernen. *Praktomat* unterstützt technisch das Kommentieren von Code, z. B. unterstützt durch studentische Hilfskräfte und so planen wir, dies zu untersuchen.

Überprüfung von studentischen Lösungen. Ähnlich wie in unserem Ansatz vergleichen auch andere Software-Werkzeuge die Lösung der studentischen SQL-Anfrage mit einer Musterlösung. Es gibt auch Ansätze, die auf die Äquivalenz der Anfragen selbst prüfen (z. B. SQLify [29]), ein Entscheidungsproblem, das nur das Vergleichen ein-

facher, *Conjunctive Queries* erlaubt. Eine entsprechende Erweiterung für *Praktomat* ist vorstellbar.

Als aus didaktischer Sicht besonders vielversprechend sehen wir das Generieren von erklärenden Beispielen an. In den aktuellen Forschungsarbeiten zu diesem Thema sehen wir großes Potential für die Datenbanklehre. Nicht zuletzt wurden die von uns generierten Rückmeldungen in der Lehrevaluation als zu wenig hilfreich moniert.

Fazit. Insgesamt haben wir mit unseren Erweiterungen von *Praktomat* eine technische Plattform geschaffen, mit der bereits jetzt die Datenbanklehre sehr gut unterstützt werden kann. Zudem hat sich das System als geeignete Basis für die Forschung zum Nutzen von E-Learning erwiesen und soll daher auch für weiterführende Untersuchungen herangezogen werden.

Danksagung Wir danken Prof. Dr. Christian Bachmaier von der Universität Passau für seine wertvollen Hinweise zum erfolgreichen Einsatz von *Praktomat*. Wir danken auch den Gutachtern für ihr detailliertes Feedback.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden.

Die in diesem Artikel enthaltenen Bilder und sonstiges Drittmaterial unterliegen ebenfalls der genannten Creative Commons Lizenz, sofern sich aus der Abbildungslegende nichts anderes ergibt. Sofern das betreffende Material nicht unter der genannten Creative Commons Lizenz steht und die betreffende Handlung nicht nach gesetzlichen Vorschriften erlaubt ist, ist für die oben aufgeführten Weiterverwendungen des Materials die Einwilligung des jeweiligen Rechteinhabers einzuholen.

Weitere Details zur Lizenz entnehmen Sie bitte der Lizenzinformation auf <http://creativecommons.org/licenses/by/4.0/deed.de>.

Literatur

1. Strecker S, Kundisch D, Lehner F, Leimeister JM, Schubert P (2018) Higher education and the opportunities and challenges of educational technology. *Bus Inf Syst Eng* 60(2):181–189
2. Zeller A (1999) Funktionell und verständlich programmieren – so lernen es die Passauer. *Softwaretech Trend* 19(3):29–34
3. Dobler H, Ramler R, Wolfmaier K (2007) A study of tool support for the evaluation of programming exercises. In: *Proc. EUROCAST*, S 376–383
4. Hass B, Yuan C, Li Z (2019) On the automatic assessment of learning outcome in programming techniques. In: *Proc. ISKE*, S 274–278
5. Müller O, Garmann R, Rod O (2019) Systeme zur automatisierten Bewertung von Programmen und das ProFormA-Aufgabenaustauschformat. In: *Teaching Trends 2018. Die Präsenzhochschule und die digitale Transformation*, S 195–200
6. Kessler J, Tschuggnall M, Specht G (2019) Relax: a webbased execution and learning tool for relational algebra. In: *Proc. BTW, LNI*, S 503–506
7. Mühe H (2012) IRA – Interaktive Relationale Algebra. Website, TU München. <https://db.in.tum.de/~muehe/ira/> Zugegriffen: 10. Dez. 2020
8. Yang J (2019) RA (radb): a relational algebra interpreter over relational databases. <https://github.com/junyang/radb>. Zugegriffen: 10. Dez. 2020
9. Scherzinger S (2019) Build your own SQL-on-hadoop query engine: a report on a term project in a master-level database course. *SIGMOD Rec* 48(2):33–38
10. Scherzinger S (2019) Have your students build their own mini hive in just eight weeks. In: *Proc. LWDA, CEUR Workshop Proceedings*, Bd. 2454, S 38–41
11. Rakow TC, Faeskorn-Woyke H, Schiefer B, Vossen G, Wäsch J (2009) Tools für die Lehre im Fach Datenbanken. *Datenbank Spektrum* 9(29):5–13
12. Obivonwu V, Broneske D, Hawiltschek A, Köppen V, Saake G (2020) SQLValidator – An Online Student Playground to Learn SQL. *Tech. rep.* <https://propra14.iti.cs.ovgu.de/sqlvali/index.php>. Zugegriffen: 22. Dez. 2020
13. Miao Z, Chen T, Bendeck A, Day K, Roy S, Yang J (2020) I-rex: an interactive relational query explainer for SQL. *Proc VLDB Endow* 13(12):2997–3000
14. Miao Z, Roy S, Yang J (2019) Explaining wrong queries using small examples. In: *Proc. SIGMOD*, S 503–520
15. Müller T, Grust T (2015) Provenance for SQL through Abstract Interpretation: Value-less, but Worthwhile. *Proc VLDB Endow* 8(12):1872–1875
16. O’Grady D, Müller T, Grust T (2018) How ‘how’ explains what ‘what’ computes – how-provenance for SQL and query compilers. In: *Proc. TaPP*
17. Bott OJ, Fricke P, Priss U, Striwe M (2017) Automatisierte Bewertung in der Programmierausbildung. Waxmann,
18. Krinke J, Störzer M, Zeller A (2002) Web-basierte Programmierpraktika mit Praktomat. *Softwaretech Trend* 22(3): S 6
19. Breitner J, Hecker M, Snelting G (2017) Der Grader Praktomat. In: *Automatisierte Bewertung in der Programmierausbildung, Digitale Medien in der Hochschullehre*, S 159–172
20. Brass S, Goldberg C (2006) Semantic errors in SQL queries: a quite complete list. *J Syst Softw* 79(5):630–644
21. Taipalus T, Perälä PMH (2019) What to expect and what to focus on in SQL query teaching. In: *Proc. SIGCSE*, S 198–203
22. Kruse M, Jensen N (2013) Automatische Bewertung von Datenbankaufgaben unter Verwendung von LON- CAPA und Praktomat. In: *Proc. ABP, CEUR Workshop Proceedings*
23. Widom J (2014) Database Mini-Courses. Online Kurs. <https://cs.stanford.edu/people/widom/DB-mooc.html>. Zugegriffen: 10. Dez. 2020
24. Kirkpatrick DL (1994) Evaluating training programs: the four levels. Berrett-Koehler,
25. Scholz M, Lehner F, Dorner V (2014) A Respecification of the DeLone and mclean model to measure the success of an electronic mediated learning systems. In: *Proc. MKWI*, S 805–819
26. Delone WH, McLean ER (2003) The delone and mclean model of information systems success: a ten-year update. *J Manag Inf Syst* 19(4):9–30
27. Lin HF (2007) Measuring online learning systems success: applying the updated delone and mclean model. *Cyberpsychology Behav* 10(6):817–820
28. Bloom B, Krathwohl D, Masia B (1956) Taxonomy of educational objectives: the classification of educational goals Bd. 1
29. Dekeyser S, de Raadt M, Lee TY (2007) Computer assisted assessment of SQL query skills. *Proc Adc CRPIT* 63:53–62
30. Do Q, Agrawal R, Rao D, Gudivada V (2014) Automatic generation of SQL queries. In: *Proc. ASEE Annual Conference and Exposition*
31. Bidgood J, Merrill JB (2017) As computer coding classes swell, so does cheating. *New York Times*. <https://www.nytimes.com/2017/05/29/us/computer-science-cheating.html>. Zugegriffen: 10. Dez. 2020
32. Prechelt L, Malpohl G, Philippsen M (2002) Finding plagiarisms among a set of programs with JPlag. *J Univers Comput Sci* 8(11):1016
33. Russell G, Cumming A (2005) Online assessment and checking of SQL: detecting and plagiarism. In: *Proc. TLAD*, S 46–50
34. Scerbakov N, Schukin A, Sabinin O (2017) Plagiarism detection in SQL student assignments. In: *Proc. ICL*, S 321–326
35. Kleerekoper A, Schofield A (2019) The False-Positive Rate of Automated Plagiarism Detection for SQL Assessments. In: *Proc. UKICER*, S 6:1–6:6
36. Kratzke N (2019) Smart like a fox: how clever students trick dumb automated programming assignment assessment systems. In: *Proc. CSEDU*, S 15–26