Students' Learning Challenges in Database Courses: A Survey Study in Information Technology Education

Sitti Suhada[1]*, Syahrul[2], Ahmad Rifqi Asrib[3], Sanatang[4]
1*, Doctoral Program in Engineering Vocational Education, Postgraduate Program, Makassar State University, Makassar, Indonesia
2,3,4Department of Faculty of Engineering, Makassar State University, Makassar, Indonesia

ABSTRACT
Database literacy sits at the core of modern IT education, yet how students actually experience the database course, and where they stumble, remains underexplored at the institutional level. This paper reports findings from a survey of 56 undergraduate students enrolled in the database course at the Department of Informatics Engineering, Universitas Negeri Gorontalo, Indonesia, drawn from two study programs: Information Technology Education (PTI, n = 27) and Information Systems (SI, n = 29). An open-ended questionnaire was used; responses were analyzed through qualitative content analysis and coded into thematic clusters. Six categories emerged. Most students (60.7%) navigated the course without major friction, while the remaining 39.3% reported concrete difficulties: Entity Relationship Diagram and normalization confusion stood out at 14.3%, followed by tool and SQL environment issues (8.9%), instructional quality gaps (8.9%), a felt disconnect between theory and hands-on work (5.4%), and hardware access limitations (1.8%). Taken together, the data suggest that while the course works well for the majority, a non-trivial minority needs targeted support, particularly around conceptual modeling and teaching responsiveness.
*Keywords:*Database learning Student challenges ERD SQL tools IT education Survey

INTRODUCTION

Ask any IT department chair which course generates the most student complaints, and the database course will surface reliably. That reputation is not undeserved. Database systems feed nearly every piece of production software in use today, from hospital record platforms to ride-hailing backends, which is why SQL proficiency and relational design skills appear on virtually every IT job posting [1]. Yet despite that practical urgency, the course consistently ranks among the tougher undergraduate offerings, placing conceptual and technical hurdles in front of students who are still building their programming intuition [2].

What students are asked to do is genuinely non-trivial. A complete database education spans conceptual modeling with Entity Relationship Diagrams (ERD), relational schema design, normalization theory through to Boyce-Codd Normal Form, and hands-on work in SQL environments like MySQL, PostgreSQL, or phpMyAdmin [3]. Each of these sits at a different cognitive register: some demand formal logical reasoning, others demand tool fluency, and a few demand both at once. The intersection is where difficulty accumulates [4].

Documented trouble spots in the computing education literature are specific. ERD construction and normalization consistently attract the most difficulty reports. Students tasked with translating a messy real-world scenario into a clean relational schema often discover that their mental model and the database model do not agree [5]. The jump from understanding a concept to running it against actual data in a DBMS adds another layer: error messages in most SQL tools are not written for beginners, and debugging a schema error with cryptic output is a different skill entirely from understanding normalization rules [6].

Indonesian higher education adds its own wrinkle. Database courses here tend to arrive in the second or third semester, before most students have meaningful programming exposure. That

[1]* Corresponding author.
E-mail address: sittisuhada@student.unm.ac.id

sequencing means instructors cannot assume the computational thinking habits that make relational logic feel intuitive [7]. Compound this with the classroom heterogeneity typical of Indonesian IT programs, where students from vocational SMK backgrounds arrive with some database exposure while their peers from general senior high schools are meeting these concepts for the first time, and you have a room with genuinely different starting points [8].

Universitas Negeri Gorontalo (UNG), situated in Gorontalo Province, runs the database course in two programs under the Department of Informatics Engineering: Information Technology Education (PTI) and Information Systems (SI). Both programs treat it as a core requirement. What has been missing is a systematic picture of what students in these specific programs actually find difficult, as opposed to what instructors assume they find difficult.

That gap matters. Instructional redesign built on assumption rather than evidence tends to fix the wrong things [9]. Survey-based diagnostic studies are a practical corrective: they capture student-perceived barriers directly and at scale, before those barriers become performance failures [10].

This study set out to build that picture for UNG's database course. Three questions organized the inquiry: What categories of difficulty do students report? How prevalent is each category? And do patterns differ between PTI and SI students? The intent is not merely descriptive; the findings feed directly into proposed instructional adjustments, and they contribute a concrete data point to the broader literature on computing education challenges in developing-country higher education.

2. METHODS

2.1 Research Design

The study used a descriptive survey design. No experimental conditions were imposed. The goal was to document student-perceived challenges as they actually existed during normal course operation [11].

2.2 Participants

Fifty-six students took part: 27 from the Information Technology Education program (PTI) and 29 from Information Systems (SI). All had either completed or were actively taking the database course at the time of data collection. Participation was voluntary; names and student IDs were collected solely for internal cross-validation and were anonymized before analysis. Purposive sampling was used to ensure all respondents had direct experience with the course content, which is a basic validity requirement for this kind of perceived-difficulty study [12].

2.3 Data Collection Instrument

An online questionnaire via Google Forms carried the data collection. Alongside demographic items (name, student ID, study program, department), participants answered one open-ended question: *"In your opinion, are there any difficulties, challenges, or problems in the database course that you have experienced so far? If yes, please explain and describe them."* Open-ended format was chosen deliberately. Closed-ended options would have constrained students to researcher-anticipated categories, potentially missing difficulty types specific to the UNG context [13].

2.4 Data Analysis

Qualitative content analysis was applied to all 56 responses, following Mayring's established framework [14]. The process ran in three stages: full reading of all responses to establish a holistic sense of the data; inductive first-cycle coding against the primary challenge theme in each response; and axial coding to consolidate codes into thematic categories. Six categories emerged: no significant difficulty, ERD and normalization challenges, programming and tools challenges, theory-practice gap, instructional quality concerns, and facility and resource constraints. Frequencies and percentages were calculated per category to support descriptive comparison across programs.

3. RESULT AND DISSCUSSION

3.1 Overall Distribution of Student Challenges

Fifty-six responses yielded six distinct patterns, as summarized in Table 1 and illustrated in Figure 1. Figure 2 further visualizes the overall split between students who reported no difficulty and those who identified specific challenges.

Table 1. Distribution of Student-Reported Challenges in the Database Course (N = 56)

No.	Challenge Category	n	Percentage (%)
1	No significant difficulty	34	60.7
2	ERD and data normalization	8	14.3
3	Programming and database tools	5	8.9
4	Instructional quality concerns	5	8.9
5	Theory-practice gap	3	5.4
6	Facility and resource constraints	1	1.8
	Total	**56**	**100.0**

Source: Primary data analysis (2025)

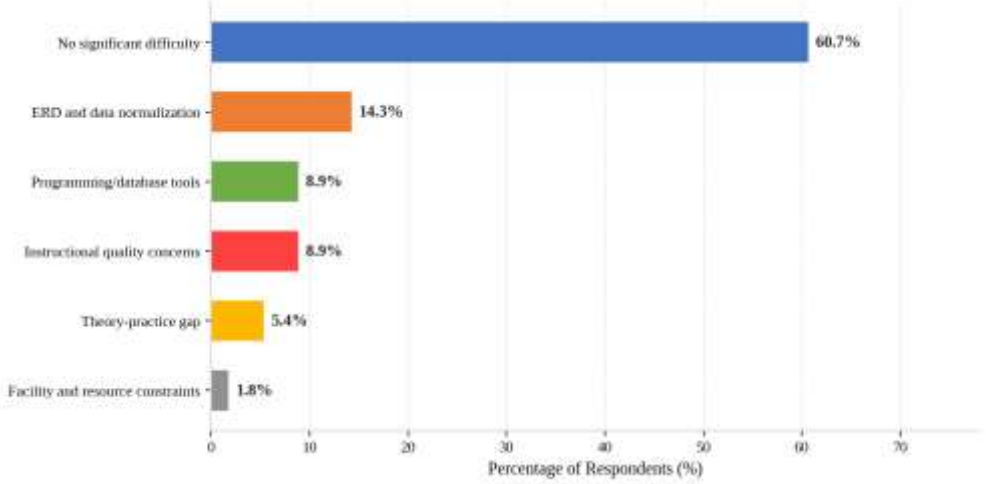

Figure 1. Percentage distribution of student-reported challenge categories in the database course (N = 56)

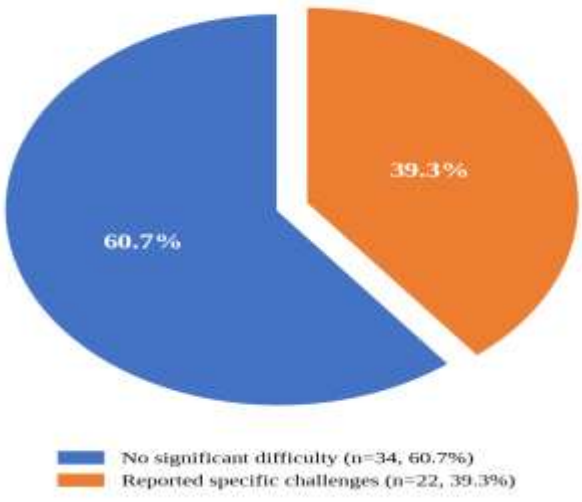

Figure 2. Proportion of students reporting no significant difficulty versus specific challenges

3.2 No Significant Difficulty (60.7%)

Thirty-four students, just over three in five, came through the course reporting no meaningful friction. Their language was direct: one wrote that course materials were "very helpful in building my understanding, so assignments and examinations could be completed well and on time"; another offered a flat "I have no difficulties, challenges, or problems." For this group, the instructional approach appears to be doing its job.

Ahadi et al. [15] offer a useful frame here: introductory computing courses with coherent explanatory structures reliably produce positive outcomes for the majority. The 60.7% figure at UNG fits that pattern. Worth noting, though, is the distorting effect that social desirability bias can introduce into self-report surveys in academic settings [16]. Students who feel reluctant to criticize a course they are still enrolled in may understate difficulties, which means the 39.3% who did report problems likely represent a genuine floor, not an inflated count.

3.3 ERD and Data Normalization Challenges (14.3%)

Eight students struggled most with ERD construction and normalization, the single largest specific-difficulty category according to Table 1. What they described was not vague confusion but a precise failure mode: taking a written system description and producing a valid relational model from it. Deciding which relationships should be one-to-many versus many-to-many, and which attributes belong on which entity, proved particularly elusive. One respondent listed relational database concepts, primary keys, foreign keys, normalization, and ERD as a cluster of interrelated abstractions that landed as "quite abstract," especially without prior programming or formal logic background.

None of this surprises computing education researchers. Mitrovic [17] pinpointed conceptual schema design as the hardest hurdle in introductory database work: the cognitive demand of converting informal, ambiguous natural language into a precise formal structure is genuinely high. Sadiq et al. [3] drill down further, noting that normalization specifically trips students up because functional dependency rules require simultaneous application and judgment, two cognitive operations that novices struggle to run in parallel.

Passive lecture delivery makes this worse. ERD and normalization both yield to active engagement through worked examples, immediate feedback on student-drawn schemas, and guided problem-solving with real datasets [5]. Intelligent tutoring systems designed for database modeling have produced measurable gains [18]; even simpler interventions, such as asking students to critique and correct deliberately flawed ERDs, show consistent benefit.

3.4 Programming and Database Tools Challenges (8.9%)

Five students ran into trouble not with the concepts but with the software environment. Three distinct friction sources appeared: transitioning from the visual, browser-based phpMyAdmin interface to command-line MySQL; XAMPP installation and configuration problems; and SQL error messages that gave little guidance on what had actually gone wrong. One student had started the course through GUI-based database interaction and found command-line work disorienting, though instructor support eventually resolved it.

That last point about error messages is well-established in the literature. Taipalus [6] showed that SQL DBMS error output is routinely opaque to beginners, generating frustration out of proportion to the actual severity of the mistake. When a student types a syntactically incorrect query and receives an error referencing internal parser state rather than explaining what is missing, debugging becomes a discouraging guessing game rather than a learning experience [2].

The fix is sequencing. Scaffolded DBMS introduction starting with GUI tools and progressing to command-line environments, with each stage explicitly connected to the

previous one, lowers the cognitive transition cost [19]. Web-based SQL environments such as SQLiteOnline or db-fiddle eliminate installation obstacles entirely and provide immediate feedback in a forgiving sandbox, which is valuable for students without reliable personal hardware [20].

3.5 Instructional Quality Concerns (8.9%)

Five respondents, the same count as the tools group, raised concerns about how the course was taught rather than what was taught. Three specific patterns appeared in their responses: instructors perceived as minimally engaged or indifferent; heavy reliance on Zoom-based synchronous delivery without compensating in-person contact; and lesson pacing that moved faster than student comprehension allowed. One respondent described being effectively told to figure things out independently, with no structured scaffolding to make self-directed learning viable.

Instructor engagement is not a soft variable. Porter et al. [7] established its role as a determinant of computing course outcomes with some precision, and Umbach and Wawrzynski [22] linked low faculty-student interaction directly to reduced academic self-efficacy and higher dropout intention in technical disciplines. The post-pandemic return to in-person instruction has, somewhat ironically, sharpened student sensitivity to engagement quality, as students who weathered fully online semesters now notice engagement gaps more acutely [21].

Structural responses matter here. Peer mentoring programs, clearly scheduled and genuinely available office hours, and asynchronous Q&A channels can absorb learning questions that synchronous class time misses [9]. Regular student feedback collection through brief mid-semester pulse surveys rather than end-of-term evaluations alone gives instructors corrective information while there is still time to act on it.

3.6 Theory-Practice Gap (5.4%)

Three students named the theory-practice gap explicitly, with a clarity that deserves quoting: one wrote that "if theory is not accompanied by practice, students do not know how to use or process databases"; another flagged "too much theory and insufficient practice." A third described learning database concepts that, in isolation from any application context, simply did not stick.

This is a curriculum design problem as much as a pedagogy problem. Connolly and Begg [1] have argued for years that database courses must weave together conceptual, logical, and physical design work with hands-on DBMS practice. Treating them as separable stages is the error that produces declarative knowledge without procedural competence [4]. When students never see their ERD turned into actual tables, populated with real data, and queried through SQL, the ERD remains an abstraction rather than a tool.

Project-based learning addresses this directly. Assign students an authentic system such as a course enrollment database, a small inventory system, or a community contact management application, and require them to move from initial requirements gathering through ERD design to physical MySQL implementation. The integration is mandatory; the theory earns its place by being immediately useful [23].

3.7 Facility and Resource Constraints (1.8%)

One student reported having no personal laptop, therefore having no independent practice outside of scheduled lab sessions, which points to an equity dimension that percentage figures tend to flatten. Database learning without personal hardware access means a student can only practice during supervised lab time. Miss a session, encounter a scheduling conflict, or need additional drill time between classes, and that student is simply out of options.

Web-based SQL platforms partially address this. A Chromebook, a smartphone with a keyboard, or any library computer with a browser can run SQLiteOnline or equivalent tools without installation [20]. Laptop lending schemes and extended lab hours are complementary

institutional responses: not high-cost interventions, but ones that remove a structural barrier rather than asking the student to work around it.

3.8 Comparative Pattern Across Study Programs

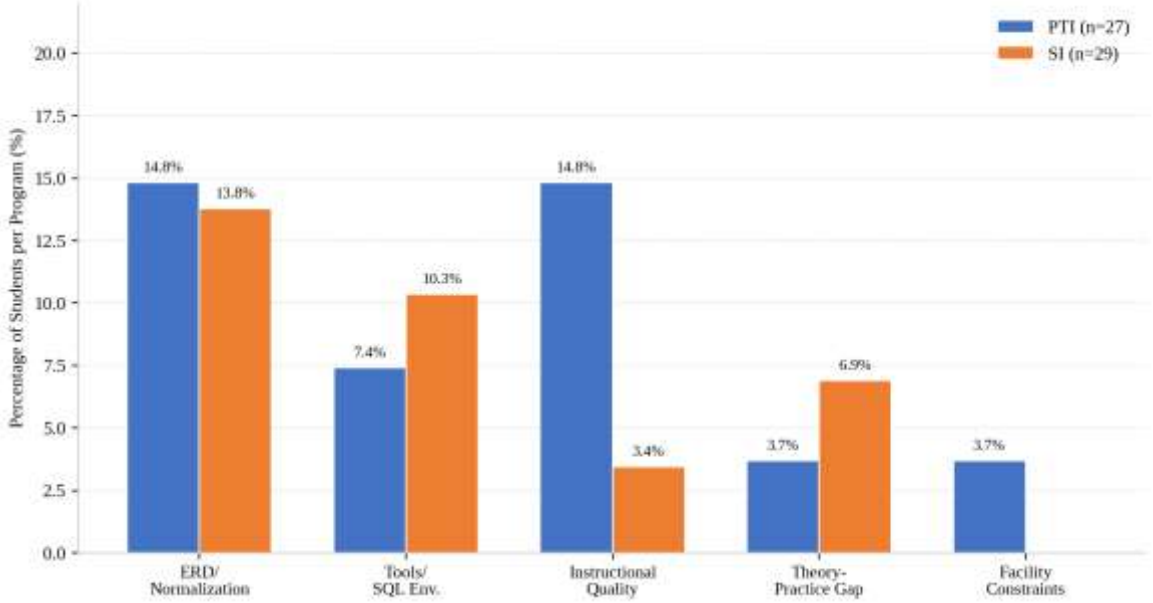

Figure 3. Estimated distribution of specific challenge categories by study program (PTI and SI)

Figure 3 displays the estimated challenge distribution across both programs. ERD and normalization difficulties were evenly distributed between PTI (14.8%) and SI (13.8%) students. Instructional quality concerns were more prominent among PTI respondents (14.8%) than among SI students (3.4%), while tool-related challenges were relatively more common in SI (10.3%) than in PTI (7.4%). The theory-practice gap was more visible in SI (6.9%) than in PTI (3.7%). These patterns, while based on small cell counts and requiring cautious interpretation, suggest that course delivery differences across sections may partly explain variation in challenge profiles.

3.9 Pedagogical Implications

Pulling the threads together from Table 1 and the figures above, five priorities emerge. First, ERD and normalization need active, corrective pedagogy: not more lecture time, but worked examples and immediate feedback on student schema attempts [17, 18]. Second, tool introduction needs explicit scaffolding from GUI to command-line, with accessible web-based alternatives for students without reliable hardware [6, 20]. Third, instructional responsiveness needs monitoring mechanisms that catch engagement gaps mid-semester rather than in end-of-term evaluations [22]. Fourth, course design needs project assignments that make the theory-practice connection unavoidable rather than incidental [23]. Fifth, lab access needs to be genuinely available for students who cannot practice independently [24].

None of these are novel recommendations. What this dataset adds is localized evidence that they are not hypothetical concerns at UNG. They are present, measurable, and, for the affected students, real obstacles.

4. CONCLUSION

Fifty-six students. Six categories. One reasonably clear picture.

The database course at UNG's Department of Informatics Engineering is working for most students, as 60.7% reported no meaningful friction. That is a reasonable baseline. What concerns the remaining 39.3% is concentrated and specific: ERD and normalization account for the largest share of genuine difficulty (14.3%), followed by tool environment challenges

and instructional quality concerns (8.9% each), a theory-practice disconnect (5.4%), and hardware access limitations (1.8%).

The implication is not that the course needs wholesale redesign. It needs targeted reinforcement at three points: earlier and more interactive engagement with conceptual modeling, deliberate scaffolding for DBMS tool transitions, and monitoring mechanisms that surface instructional quality gaps before they become performance gaps. Project-based assessments that require physical implementation alongside conceptual design would address both the theory-practice complaint and the normalization confusion simultaneously.

This study contributes a located, empirical data point to the literature on database education challenges in Indonesian higher education. Its scope is limited: one department, one survey instrument, one snapshot in time. Future work should pair survey data with performance outcomes to test whether reported difficulties predict grade patterns, and should follow cohorts longitudinally to track how early course challenges propagate into later technical subjects. Intervention studies testing the specific pedagogical responses outlined above would close the loop between diagnosis and solution.

ACKNOWNLEDGMENT

The author gratefully acknowledges the students of the Department of Informatics Engineering, Universitas Negeri Gorontalo, who voluntarily participated in this survey study and Universitas Negeri Makassar for its support throughout the course of this research.

FUNDING

This research received no external funding.

AUTHOR CONTRIBUTION

Conceptualization, S.S., S., A.R.A. and S.,; methodology, S.; software, S.S and S.; validation, S..; formal analysis, S.S.; investigation, S.S.; resources, S.S.; data curation, S.S.; writing, original draft preparation, S.S.; writing, review and editing, S.S. and S.,; visualization, S.S.; supervision, S. All authors have read and agreed to the published version of the manuscript.

CONFLICT OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] [1] Connolly, Thomas M., and Carolyn E. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed. Harlow: Pearson Education, 2014. ISBN 978-0-13-294326-0.

[2] Taipalus, Toni, Mikko Siponen, and Tero Vartiainen. "Errors and Complications in SQL Query Formulation." *ACM Transactions on Computing Education* 18, no. 3 (2018): 1-29. https://doi.org/10.1145/3231712

[3] Sadiq, Shazia, Sourav Bhowmick, and Graham Brown. "Identifying Difficulties in Learning Conceptual Data Modeling." In *Proceedings of the 16th Australasian Database Conference*, 105-113. Darlinghurst: Australian Computer Society, 2004.

[4] Smelcer, John B. "User Errors in Database Query Composition." *International Journal of Human-Computer Studies* 42, no. 4 (1995): 353-381. https://doi.org/10.1006/ijhc.1995.1017

[5] Mannila, Linda, and Mirja Peltomaki. "It Is Just a Program, Nothing More or Less: Students' Conceptions of Computing." In *Proceedings of the 6th Baltic Sea Conference*

on Computing Education Research, 101-108. New York: ACM, 2006. https://doi.org/10.1145/1315803.1315821

[6] Taipalus, Toni. "Database Management System Error Messages for Novice Users." In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 284-290. New York: ACM, 2020. https://doi.org/10.1145/3341525.3387412

[7] Porter, Leo, Mark Guzdial, Charlie McDowell, and Beth Simon. "Success in Introductory Programming: What Works?" *Communications of the ACM* 56, no. 8 (2013): 34-36. https://doi.org/10.1145/2492007.2492020

[8] Surjono, Herman Dwi, Ali Muhtadi, and Dian Wahyuningsih. "The Development of Interactive Multimedia-Based E-Learning for Computer Network Course." *Journal of Educational Technology and Online Learning* 1, no. 1 (2017): 1-12. https://doi.org/10.31681/jetol.383556

[9] Prather, James, Raymond Pettit, Kayla McMurry, Amey Peters, John Homer, and Mike Cohen. "On the Difficulties of Learning to Program." In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, 1-10. New York: ACM, 2018. https://doi.org/10.1145/3279720.3279741

[10] Ko, Andrew J., Thomas D. LaToza, and Margaret M. Burnett. "A Practical Guide to Controlled Experiments of Software Engineering Tools with Human Participants." *Empirical Software Engineering* 20, no. 1 (2015): 110-141. https://doi.org/10.1007/s10664-013-9279-3

[11] Creswell, John W., and J. David Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 5th ed. Thousand Oaks: SAGE Publications, 2018. ISBN 978-1-5063-8670-6

[12] Patton, Michael Q. *Qualitative Research and Evaluation Methods*. 4th ed. Thousand Oaks: SAGE Publications, 2014. ISBN 978-1-4129-7212-3.

[13] Denzin, Norman K., and Yvonna S. Lincoln, eds. *The SAGE Handbook of Qualitative Research*. 5th ed. Thousand Oaks: SAGE Publications, 2017. ISBN 978-1-4833-4980-0.

[14] Mayring, Philipp. "Qualitative Content Analysis." *A Companion to Qualitative Research* 1, no. 2 (2004): 159-176.

[15] Ahadi, Alireza, Vahid Behbood, Aman Yadav, Jamie Fraser, and Raymond Lister. "Students' Difficulties with Recursion, and Hints to Make It Easier." In *Proceedings of the 17th Australasian Computing Education Conference*, 162-170. Darlinghurst: Australian Computer Society, 2015.

[16] Fisher, Robert J. "Social Desirability Bias and the Validity of Indirect Questioning." *Journal of Consumer Research* 20, no. 2 (1993): 303-315. https://doi.org/10.1086/209351

[17] Mitrovic, Antonija. "Learning SQL with a Computerized Tutor." *ACM SIGCSE Bulletin* 30, no. 1 (1998): 307-311. https://doi.org/10.1145/274790.274463

[18] Mitrovic, Antonija, Brent Martin, Pramuditha Suraweera, Konstantin Zakharov, Neil Milik, James Holland, and Nigel McGuigan. "ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors." *International Journal of Artificial Intelligence in Education* 19, no. 2 (2009): 155-188.

[19] Vygotsky, Lev S. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge: Harvard University Press, 1978. ISBN 978-0-674-57629-2.

[20] Leinonen, Juho, Kalle Longi, Arto Klami, and Arto Hellas. "Automatic Inference of Programming Performance and Experience from Typing Patterns." In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 132-137. New York: ACM, 2016. https://doi.org/10.1145/2839509.2844612

[21] Watermeyer, Richard, Tom Crick, Cathryn Knight, and Janet Goodall. "COVID-19 and Digital Disruption in UK Universities: Afflictions and Affordances of Emergency Online Migration." *Higher Education* 81, no. 3 (2021): 623-641. https://doi.org/10.1007/s10734-020-00561-y

[22] Umbach, Paul D., and Matthew R. Wawrzynski. "Faculty Do Matter: The Role of College Faculty in Student Learning and Engagement." *Research in Higher Education* 46, no. 2 (2005): 153-184. https://doi.org/10.1007/s11162-004-1598-1

[23] Krajcik, Joseph S., and Namsoo Shin. "Project-Based Learning." In *The Cambridge Handbook of the Learning Sciences*, edited by R. Keith Sawyer, 275-297. 2nd ed. Cambridge: Cambridge University Press, 2014. https://doi.org/10.1017/CBO9781139519526.018

[24] Warschauer, Mark, and Tina Matuchniak. "New Technology and Digital Worlds: Analyzing Evidence of Equity in Access, Use, and Outcomes." *Review of Research in Education* 34, no. 1 (2010): 179-225. https://doi.org/10.3102/0091732X09349791