ISSN: 2473-4901
v10 n6171

# Assessing the Impact of Prerequisite Courses on Student Performance in Relational Design

Kevin Slonka
kslonka@francis.edu
Computer Science & Cyber Security Department
Saint Francis University
Loretto, PA 15940, USA

Matthew North
mnorth@uvu.edu
Information Systems & Technology Department
Utah Valley University
Orem, UT 84058, USA

Neelima Bhatnagar
bhatnagr@pitt.edu
Information Sciences Department
University of Pittsburgh
Greensburg, PA 15601, USA

Anthony Serapiglia
anthony.serapiglia@stvincent.edu
CIS Department
Saint Vincent College
Latrobe, PA 15650, USA

## Abstract

Continuing to fill the literature gap, this research replicated and expands a prior study of student performance in relational design in an introductory database course. The data was collected from four different universities, each having different prerequisite courses for their database course. Student performance on a relational design exam was compared between students who took the various studied prerequisite courses (CS1 procedural programming, CS2 object-oriented programming, and discrete mathematics) and those who did not. These comparisons were conducted using quantitative methods and produced non-significant results, different from those results of the prior study. With the current research design and sample size, the researchers are confident in their results that suggest these prerequisite courses do not impact student performance.

**Keywords:** database, normalization, programming, discrete math, pedagogy

## 1. INTRODUCTION

One of the core concepts taught in introductory database courses is relational modeling. Despite its importance, students encountering this concept for the first time often struggle with both comprehension and application (Bunch, 2009). These challenges often stem from a lack of prior exposure to key concepts such as data normalization, cardinality, and keys. The abstract nature of relational databases further complicates the learning process for new students (Freyberg, 1996).

Common problems for students new to relational databases include difficulty in grasping the logical structure of data relationships, confusion over the syntax and semantics of coding languages such as Structured Query Language (SQL), and an inability to effectively translate real-world scenarios into database designs (Philip, 2007). Misunderstandings in foundational topics such as primary and foreign keys, normalization forms, and query optimization frequently lead to frustration and hinder academic progress. Further, students often struggle with the transition from conceptual design to implementation, a gap that is critical for practical proficiency in database management (Thompson & Sward, 2005).

Given this reality, it is reasonable to hypothesize that students with some background in computer programming logic and abstract thinking will realize higher levels of comprehension and achievement than database students who lack such background. This study tests that theory by testing one primary hypothesis constructed from the overarching research question, “What impact do prerequisite courses have on student performance in relational design”:

*$H_1$: Procedural programming experience does not affect student scores on database normalization.*

Additionally, this study added two additional hypotheses to analyze the remaining two possible prerequisite courses:

*$H_2$: Object-oriented programming experience does not affect student scores on database normalization.*

*$H_3$: Discrete math experience does not affect student scores on database normalization.*

This research will analyze and discuss the student, faculty, and organizational impact of these prerequisite courses.

## 2. REVIEW OF THE LITERATURE

The design of relational databases has been found to be a difficult concept for students to grasp (Thompson & Sward, 2005; Kung & Tung, 2006; Hingorani, Gittens, & Edwards, 2017). Further exacerbating the difficulty are the inconsistencies in notational styles for Entity-Relationship Diagrams (ERD), the most common technique for visualizing relational databases, such as Thompson & Sward’s (2005) proposed approach. Beyond the issues with diagramming is one of the core tenants of relational design, the process of normalization.

Pedagogical research into mitigating challenges and enhancing the teaching and learning of relational database concepts, such as normalization, has been conducted for at least the past 25 years (Chan, Teo, & Zeng, 2005; Philip, 2007). Early efforts to improve student comprehension of relational database concepts often assumed that students needed foundational coursework in programming and abstract math in order to understand relational modeling (Chilton, McHaney, & Bongsug, 2006; Kung & Tung, 2006; Zhang, Kaschek, & Kinshuk, 2005). This is further worsened by the disagreement of whether third normal form (3NF) is satisfactory or if further forms should be taught (e.g., 4NF, BCNF, 5NF, DKNF, etc.) (Carpenter, 2008). However, work by Dominguez & Jaime (2010) and Enciso & Soler (2013) and others soon challenged this assumption by testing methods of teaching relational database concepts through contextualization and concept attainment. Much of the early research on teaching database design focused around the question: Must students understand programming and abstraction to successfully design relational databases?

Research into cognitive load theory began to inform the structuring of course content to avoid overwhelming students with complex information prematurely (Batra & Wishart, 2014; Bunch, 2009). Empirical studies over the past 15 years have analyzed student performance in database classes to identify effective strategies and relevant foundational knowledge, guiding the continuous improvement of educational practices (Katz, 2020; Mitrovic & Suraweera, 2016). One such strategy is providing visualizations and ensuring that students have plenty of opportunities to interact (Hamzah et al., 2019; Folorunso & Akinwale, 2010; Jaimez-Gonzoalez & Martinez-Samora, 2020). Tayal & Bura (2012) successfully tested a novel approach to teaching relational design to novices through a concept attainment teaching method, finding that a plurality of students, regardless of prior coursework in computing or math, achieved at least a satisfactory level of comprehension of course content.

Sastry (2015) and, more recently, Sing et al. (2021) examined an innovative teaching method focused on problem-based learning to enhance students' understanding of core database design concepts. Their approach employed practical, hands-on problem-solving activities, rather than traditional lecture methods. By designing teaching examples and homework exercises that emphasized both critical thinking and technical skills, engaging students in real-world database design challenges, their research showed higher levels of student achievement as the learning process became more interactive. Wang & Wang (2023) found similar results by employing interactive NoSQL database instruction within business education. Their research tests the use of practical exercises and projects to help students understand concepts, data models, and query languages, resulting in improved student performance when compared to traditional teaching methods. This research background informs the experimental design in our study.

## 3. METHODOLOGY

Five sections of a database management class, in which all students were computing majors, were studied. One section was taught at a regional campus of an R1 university in Western Pennsylvania. This course did not require any prerequisite programming courses or math courses. The second section was taught at a Western Pennsylvania Catholic university that requires a procedural programming course as a prerequisite, though some students also had taken an object-oriented programming course as well as a discrete mathematics course. The third section was taught at a different Western Pennsylvania Catholic college. This particular course had prerequisites of procedural programming, object-oriented programming, and discrete mathematics. The fourth and fifth sections were taught at the largest public university in a particular Western-US state. This university's database course also did not require any prerequisite courses in programming or mathematics. The sum of all participants in this study was n=136.

The researchers coordinated their content and grading, with all courses teaching the same concepts that culminated in an exam on normalization. All sections administered a standard exam and all grading was done with a standard rubric to ensure the integrity of the study. The exam was a raw data set in a Microsoft Excel file that asked the students to normalize the data into third normal form. The instructions given to students are shown in Appendix A.

In addition to collecting the grades of each student, the researchers also gathered data on whether or not each student had taken their university's CS1 (procedural programming), CS2 (object-oriented programming), and discrete mathematics course. The collection of this data allowed for the testing of this study's primary hypotheses:

*$H_1$: Procedural programming experience does not affect student scores on database normalization.*

*$H_2$: Object-oriented programming experience does not affect student scores on database normalization.*

*$H_3$: Discrete math experience does not affect student scores on database normalization.*

## 4. RESULTS

The demographics of the population are shown below in Table 1 as well in Appendix B Figure 1. While the majority of the participants had taken the CS1 course, the numbers evened out with CS2 and discrete math due to those courses either not being a prerequisite of the database course or not being a required course in their major.

| | n | % |
|---|---|---|
| Total | 136 | 100.0 |
| CS1 | | |
| Yes | 115 | 84.6 |
| No | 21 | 15.4 |
| CS2 | | |
| Yes | 88 | 64.7 |
| No | 48 | 35.3 |
| Discrete Math | | |
| Yes | 71 | 52.2 |
| No | 65 | 47.8 |

**Table 1: Frequencies**

The data was then checked for normality to determine the proper statistical analyses to run. When viewing the scores grouped by each prerequisite course it was apparent that the data was non-normal but could be argued to be either normal or non-normal, depending on whether one focused on the "yes" answers or the "no" answers, shown in Appendix B Figure 2 with the combined histograms and Q-Q plots. Focusing only on the empiric distribution, the normality discrepancy is apparent.

The empiric nature of the data's normality is backed up by the Skewness and Kurtosis values, which can also lead one to choose either parametric or non-parametric analyses due to some values being closer to 0 than others, shown in Table 2.

| | | Skewness | Kurtosis |
|---|---|---|---|
| CS1 | Yes | -0.391 | -0.904 |
| | No | -0.117 | -0.906 |
| CS2 | Yes | -0.412 | -1.086 |
| | No | -0.643 | -0.493 |
| Discrete Math | Yes | -0.333 | -1.137 |
| | No | -0.599 | -0.181 |

**Table 2: Skewness & Kurtosis for Normality Testing**

Because of these inconsistencies, the researchers ran both the parametric and non-parametric tests. For each hypothesis the non-parametric test (Mann-Whitney U) will be presented followed by the parametric test (independent samples T-Test). Although the grouping variable only contains two groups (yes/no), the researchers also ran the stronger ANOVA test, which typically requires at least 3 groups, to be completely certain that their results were valid (Pallant, 2010).

**$H_1$: Procedural programming experience does not affect student scores on database normalization.**

Shown in Table 3, the non-parametric Mann-Whitney U test did not produce a significant result. When running the independent samples T-Test, the significance value was taken from the "Equal variances not assumed" portion of the test due to the significant Levene's Test for Equality of Variances. This test also did not produce a significant result. The stronger parametric test, the ANOVA, corroborated the previous two tests with a non-significant finding.

| Test | Significance |
|---|---|
| Mann-Whitney U | 0.818 |
| T-Test | 0.468 |
| ANOVA | 0.588 |

**Table 3: Parametric & Non-Parametric Statistics for $H_1$**

These results lead the researchers to accept the null hypothesis, suggesting that procedural programming experience does not affect student scores on database normalization.

**$H_2$: Object-oriented programming experience does not affect student scores on database normalization.**

Shown in Table 4, the non-parametric Mann-Whitney U test did not produce a significant result. When running the independent samples T-Test, the significance value was taken from the "Equal variances not assumed" portion of the test due to the significant Levene's Test for Equality of Variances. This test also did not produce a significant result. The stronger parametric test, the ANOVA, corroborated the previous two tests with a non-significant finding.

| Test | Significance |
|---|---|
| Mann-Whitney U | 0.321 |
| T-Test | 0.501 |
| ANOVA | 0.534 |

**Table 4: Parametric & Non-Parametric Statistics for $H_2$**

These results lead the researchers to accept the null hypothesis, suggesting that object-oriented programming experience does not affect student scores on database normalization.

**$H_3$: Discrete math experience does not affect student scores on database normalization.**

Shown in Table 5, the non-parametric Mann-Whitney U test did not produce a significant result. When running the independent samples T-Test, the significance value was taken from the "Equal variances assumed" portion of the test due to the non-significant Levene's Test for Equality of Variances. This test also did not produce a significant result. The stronger parametric test, the ANOVA, corroborated the previous two tests with a non-significant finding.

| Test | Significance |
|---|---|
| Mann-Whitney U | 0.519 |
| T-Test | 0.590 |
| ANOVA | 0.590 |

**Table 5: Parametric & Non-Parametric Statistics for $H_3$**

These results lead the researchers to accept the null hypothesis, suggesting that discrete math experience does not affect student scores on database normalization.

## 5. DISCUSSION

This study was a more comprehensive replication of a prior study by Slonka & Bhatnagar (2023). The prior study was limited with a small sample size and the lack of a standardized exam and grading rubric. This study corrected those errors while also expanding the sample to more universities in more geographic areas. This allowed the results of this study to be more generalizable and more legitimate.

The prior study found that having prior experience with procedural programming led to lower normalization scores and having prior discrete math experience also led to lower normalization scores. These results were unsatisfactory due to the previously mentioned errors. This study corrects those missteps by suggesting that students' grades on normalization are not affected by any of the three prerequisite courses tested. This has multiple implications for computing majors that include a database course.

First, removing these prerequisites from the database course could lead to higher enrollments in the class sections. This would not only benefit faculty, ensuring their sections do not get canceled due to low enrollment, but also the university for obvious financial reasons.

Second, students could be able to take the database course earlier in their college career. Being able to take a database course in the first or second year of a major, instead of during the last two years, would allow computing departments to strengthen their data-focused curriculum, offering higher-level database courses as well as other courses that would typically require the database course as a prerequisite, such as data analytics or data warehousing.

Third, the removal of these prerequisites could remove the barrier preventing students outside of a university's computing majors from taking the database course. Databases play a role in many non-computing majors and having this course available for non-computing majors could result in the need for many more sections, which also translates to dollar signs. Lastly, it is possible that the simpler nature of SQL as a programming language could be a gateway for students to enroll in more intense computer science courses. This could greatly improve enrollment in computing majors, but would need further research to determine legitimacy.

## 6. CONCLUSION

In investigating student comprehension of the relational database model, this study aimed to identify the impact of prior knowledge in programming concepts and discrete mathematics on learning outcomes. Our analysis reveals no statistically significant differences in comprehension levels between students with a background in these areas and those without. This finding suggests that familiarity with programming or discrete mathematics does not inherently translate to a better understanding of relational databases. The lack of a significant correlation suggests that while programming and mathematical skills are valuable, and are a core, indispensable component of college-level academic programs in the computing fields, they may not directly influence a student's ability to grasp the relational database model.

These findings underscore the necessity of developing specialized teaching methods tailored specifically to relational databases, rather than relying on students' prior knowledge in related fields. Emphasizing fundamental database concepts from the ground up, using practical examples, interactive learning tools, and active engagement techniques, may prove more effective in fostering comprehensive understanding. This study did not test the effectiveness of specific teaching methods or tools, only the effect of having taken programming or math classes before taking the introductory database course. We find that students should be encouraged to take their first database class early in their studies, whether they have already taken programming or discrete mathematics or not.

Another limitation, for which it would be difficult to control, is any exposure to database concepts the students received prior to their college career. Although the researchers' experience leads them to believe that these concepts are not found in any secondary school curriculum, it remains a possibility.

This study highlights the importance of faculty engaging in continuous pedagogical research to identify and implement teaching practices that directly address the unique challenges faced by students learning relational databases. Future research should explore a broad range of instructional techniques, including peer-assisted learning, real-world project-based assignments, and adaptive learning technologies, to enhance student comprehension and retention. Additionally, research into different visualization tools and other possible prerequisite courses could yield fruitful results.

## 7. REFERENCES

Batra, D., & Wishart, N. (2014). Novice designer performance comparison between the entity relationship event network and the event-based logical relational design techniques. *Journal of Database Management, 2*5(3), 1–27. https://doi.org/10.4018/jdm.2014070101

Bunch, J. M. (2009). An approach to reducing cognitive load in the teaching of introductory database concepts. *Journal of Information Systems Education, 20*(3), 269–275.

Carpenter, D. A. (2008). Clarifying normalization. *Journal of Information Systems Education, 19*(4), 379-382.

Chan, H. C., Teo, H. H., & Zeng, X. H. (2005). An evaluation of novice end-user computing performance: Data modeling, query writing, and comprehension. *Journal of the American Society for Information Science & Technology, 56*(8), 843–853. https://doi.org/10.1002/asi.20178

Chilton, M. A., McHaney, R., & Bongsug Chae. (2006). Data modeling education: The changing technology. *Journal of Information Systems Education, 17*(1), 17–20.

Domínguez, C., & Jaime, A. (2010). Database design learning: A project-based approach organized through a course management system. *Computers & Education, 55*(3), 1312–1320. https://doi.org/10.1016/j.compedu.2010.06.001

Enciso, M., & Soler, E. (2013). Teaching database design: A reverse engineering approach. 2013 IEEE Global Engineering Education Conference (EDUCON), *2013 IEEE*, 474–480. https://doi.org/10.1109/EduCon.2013.6530148

Folorunso, O. & Akinwale, A. (2010). Developing visualization support system for teaching/learning database normalization. *Campus-Wide Information Systems, 27*(1), 25-39. https://doi.org/10.1108/10650741011011264

Freyberg, C. A. (1996). Teaching independent novice learners to develop relational database applications. *Proceedings 1996 International Conference Software Engineering: Education and Practice, Software Engineering: Education and Practice*, 64–67. https://doi.org/10.1109/SEEP.1996.533982

Hamzah, M. L., Rukun, K., Fahmi, R., Purwati, A. A., Hamzah, H., & Zarnelly. (2019). A review of increasing teaching and learning database subjects in computer science. *Revista Espacios, 40*(26), 6-14.

Hingorani, K., Gittens, D., & Edwards, N. (2017). Reinforcing database concepts by using entity relationships diagrams (ERD) and normalization together for designing robust databases. *Issues in Information Systems, 18*(1), 148-155.

Jaimez-Gonzalez, C. R. & Martinez-Samora, J. (2020). DiagrammER: A web application to support the teaching-learning process of database courses through the creation of E-R diagrams. International *Journal of Emerging Technologies in Learning, 15*(19), 4-21. https://doi.org/10.3991/ijet.v15i19.14745

Katz, A. (2020). Improved teaching of database schema modeling by visualizing changes in levels of abstraction. *Journal of Information Systems Education, 31*(4), 294.

Kung, H. & Tung, H. (2006). An alternative approach to teaching database normalization: A simple algorithm and an interactive e-learning tool. *Journal of Information Systems Education, 17*(3), 315-325.

Mitrovic, A., & Suraweera, P. (2016). Teaching database design with constraint-based tutors. *International Journal of Artificial Intelligence in Education: Official Journal of the International AIED Society, 26*(1), 448–456. https://doi.org/10.1007/s40593-015-0084-6

Pallant, J. (2010). SPSS survival manual. McGraw Hill.

Philip, G. C. (2007). Teaching database modeling and design: Areas of confusion and helpful hints. *Journal of Information Technology Education, 6*, 481–497. https://doi.org/10.28945/228

Sastry, K. S. (2015). An effective approach for teaching database course. International *Journal of Learning, Teaching, and Educational Research, 12*(1), 53-63.

Singh, A., Bhadauria, V., & Gurung, A. (2021). A problem-solving-based teaching approach to database design. *Journal of Emerging Technologies in Accounting, 18*(2), 149–155. https://doi.org/10.2308/JETA-19-10-13-41

ISSN: 2473-4901
v10 n6171

Slonka, K. J. & Bhatnagar, N. (2023). Teaching database normalization: Do prerequisites matter? *Proceedings of the ISCAP Conference*. Albuquerque, NM: ISCAP.

Tayal, D., & Bura, D. (2012). An efficient approach for identifying functional dependencies in a database class. *IUP Journal of Information Technology, 8*(1), 36–47.

Thompson, C. B., & Sward, K. (2005). Modeling and teaching techniques for conceptual and logical relational database design. *Journal of Medical Systems, 29*(5), 513–525. https://doi.org/10.1007/s10916-005-6108-3

Wang, H., & Wang, S. (2023). Teaching NoSQL databases in a database course for business students. *Journal of Information Systems Education, 34*(1), 32.

Zhang, L., Kaschek, R., & Kinshuk. (2005). Developing a knowledge management support system for teaching database normalization. *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05), Advanced Learning Technologies*, 344–348. https://doi.org/10.1109/ICALT.2005.117

## APPENDIX A

Normalization Exam (100 pts)

Using the exam-data.xlsx file, create a normalized database (3rd normal form).

Considerations:

- There are 5 main entities within this data but you will need 7 to properly design the database.
- Students enroll in course sections, not courses.
- Students should be allowed to enroll in multiple course sections.
- Surrogate keys are allowed and are, in most cases, preferred. You must decide whether the data already contains a proper primary key or you need to create a surrogate or composite key.

Upload the following item to complete this exam:

1. ERD (PDF or image file)
    a. Diagram must use correct Crow's Foot notation. An online tool, such as Lucid Chart, is sufficient for this.

ISSN: 2473-4901
v10 n6171

## APPENDIX B

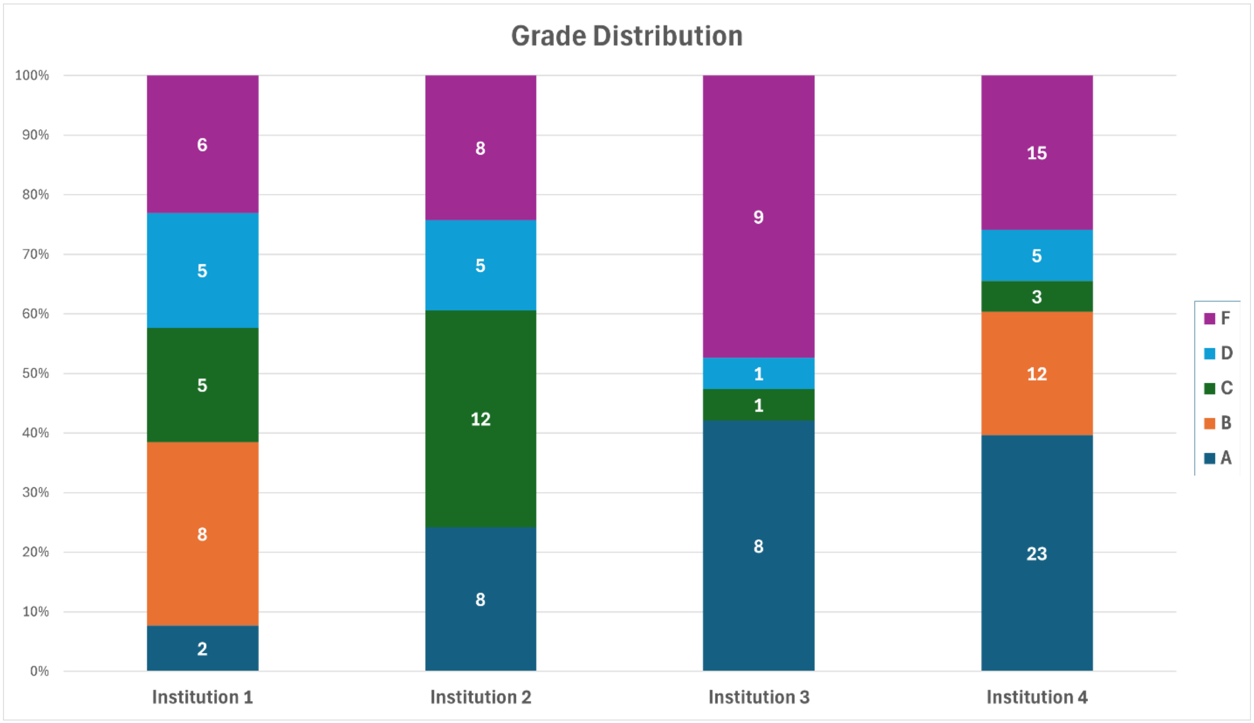

**Figure 1: Grade Distribution by Institution**

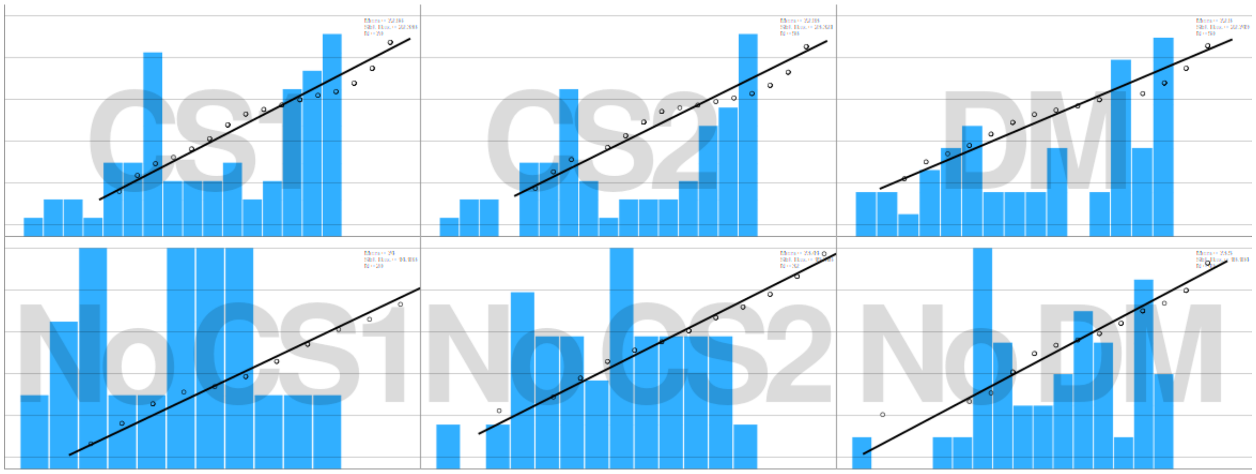

**Figure 2: Histograms and Q-Q Plots**