Int. J. Engng Ed. Vol. 24, No. 5, pp. 980–989, 2008
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Best Teaching Practices in Database Courses for Engineering Students*

ABE ZEID and SAGAR KAMARTHI
Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA 02115, USA.
E-mail: zeid@coe.neu.edu sagar@coe.neu.edu

The National Academy of Sciences recommends adapting engineering education to the new century. One of its recommendations is that engineering schools introduce interdisciplinary learning at the undergraduate level. An area that lends itself well to interdisciplinary learning is information technology (IT). Sample IT courses are Computers and Information Systems and Databases. Teaching IT courses to engineering students is more challenging than traditional engineering. Here we focus on teaching a database course in a mechanical and industrial engineering curriculum, and on the best practices to teach such a course. We make a threefold contribution. First, we discuss the obstacles to teaching IT courses in engineering. Second, we provide a model to teach database courses to engineering students. Third, we offer a blueprint to engineering educators who are interested in or thinking about introducing such courses in their curricula.

Keywords: Engineering education; IT; databases; productive pedagogy; globalization

INTRODUCTION

ENGINEERING EDUCATION has been the subject of change [1–5] to meet new demands such as globalization and interdisciplinary knowledge [6–11]. Today's engineering practice most often requires engineers to be diverse and be ready to work in engineering related fields such as nanotechnology, software and others. The National Academy of Sciences discusses Adapting Engineering Education to the new century by showing the desired Aspirations and Attributes of Engineers of 2020 [1, 2]. Some of the engineer's desired attributes include strong analytical skills, practical ingenuity, creativity, communication, business and management, dynamism, agility, resilience and flexibility and lifelong learning. Collectively, these attributes promote the interdisciplinary approach to engineering education.

The IT area can expose engineering students to some useful concepts [12]. Within IT, databases are important in modern computing and use of software. Commercial and business applications rely heavily on the use of databases, storage of massive amount of product and customer data in databases and retrieving information when needed from these databases. All the common Internet applications, Web social networks and others, use databases. Microsoft is shifting its popular Windows operating system and Office into the Internet by introducing Windows Live and Office Live services [13]. Users of these services do not install any software on their local computers. They pay per use. This paradigm of computing shifts the burden of storing data and files from the users to the providers of the computing services, thus requiring the providers to manage and maintain large databases and warehouses.

While databases are pervasive in software and related applications, they are equally pervasive in all engineering applications. Consider, for example, mechanical and industrial engineering applications. Commercial CAD/CAM software is based on relational database concepts [14–20] to store and manipulate files of parts and assemblies. Industrial engineers create databases for many applications including quality control, simulations, human factors, etc.

Anecdotal observations by the authors based on the IT courses that they teach, and the feedback they receive in professional meetings, confirm the need for engineers to learn database concepts. Northeastern University provides its students with experiential learning experience in the form of its co-op program where students alternate periods of work and study during their undergraduate education. Co-op is mandatory for engineering. Multiple students, especially industrial engineering students, speak of their co-op jobs that require databases. Sometimes, students outside the authors' department and, even the college of engineering, take their database course.

COURSE DESIGN AND CONTENT

Design constraints

Teaching database courses in computer science curricula is expected and required. However, teaching databases to engineering students is quite different. The subject is non-traditional for engineers. The database concepts are abstract, and engineering students prefer a hands-on and problem-solving approach. Moreover, there is a lack of database

* Accepted 30 March 2008.

textbooks that are engineering-oriented. Lastly, it is engineering faculty who teach such a course.

With these constraints, we set to design a database course that blends both the concepts and the practice of databases. The course is designed on the premise that the effective practice of databases and applications is grounded in understanding the basic concepts of the subject. Such a course begins by teaching the students the difference between data and information. Knowing this difference provides the students with an appreciation of why we invest time and money to create databases. They get to understand that databases store data. After they input and store the data, they can query the database to retrieve information to uncover trends and other valuable knowledge hidden in the data.

Content

The course has three main parts. The first part covers the concepts of data modeling and sound and robust database design. The main coverage includes design tools such as tables, Entity Relationship Diagrams (ERDs), relational schema, normalization and data dictionary. A table is the low level entity for storing data in a database. Each table has attributes, and each attribute is of a specific data type. ERDs define the relationships between tables. Without relationships, the integrity of a database is compromised. That is, one table may point to already deleted data in another table. Relational schema show the detailed design of each table in the database and connect its tables on the proper tables' keys. The normalization concept removes data redundancy from tables and fine tunes the table structures. The data dictionary shows the data type and format of each table attribute.

The second part of the course focuses on data mining and warehousing via Sructured Query Language (SQL). SQL enables us to extract information and trends from databases. The user writes an SQL query and executes it against the database, as a statement or set of statements, to retrieve the desired information. Here, students learn the basic syntax of SQL and the available types of SQL statements and when to use each type. We break the SQL syntax into two groups: unconditional SQL statements and conditional ones. We cover the different types of statements within each group.

The third part of the course focuses on the use of commercial database software for two purposes. First, we use the database software to illustrate the abstract concepts throughout the course. Second, students use the software to fulfill the project requirements of the course. There are two options when it comes to selecting software: Windows-based or open-source Linux/Unix-based. Students are given the choice of selecting one option. The entire class must select only one option because we cannot support two options simultaneously in the course. The Windows option uses Microsoft Access software which is readily available through campus computer labs. The Linux/Unix option uses MySQL and PHP programming language. Engineering students always select Access for two reasons. First, they state that Access is used more by their co-op employers. Second, engineering students, unlike computer science students, do not like programming in general. Their only exposure to programming is taking a Matlab/C++ programming course during their freshman year. They usually take the database course during their third year.

Textbook and class material

Once the course design and content is finalized, the next challenging task is to identify and select an appropriate textbook for the course. After exhaustive search, we could not find a book that is a perfect match for what we want to teach. The majority of existing database textbooks fall into two groups. One group includes textbooks that are highly abstract. These are suitable for computer science majors. The other group includes textbooks that focus primarily on teaching the syntax and use of a given software such as Access. Neither of these groups fits the philosophy of our course. Nevertheless, we use the textbook by Rob and Semaan [16]. We supplement the book by many of our own notes, material and examples.

Course syllabus

The outcome of the above design constraints and content is a four-hour semester engineering database course offered by our department. The course does not have any prerequisites. The course details are as follows:

- **Course Description.** Examines the representation of data and its creation and management in engineering enterprises. Discusses the client/server model of database access. Presents the fundamentals of data modeling and management, data mining and warehousing, multitier applications and the use of the SQL query language. Emphasizes the use and applications of database systems in engineering including design and manufacturing. Topics include design schema of tables, records and fields of databases, and SQL statements.
- **Course Objectives.** The Students are expected to learn and demonstrate the following abilities:
  1. To know the difference between data and information
  2. To understand ERDs for data modeling
  3. To understand the client/server model of database access
  4. To understand database design schema of tables, records, and fields
  5. To perform data mining and warehousing using SQL queries.
- **Course Topics.**
  1. Database concepts and design tools
  2. Relational schema
  3. Normalizing database table structures
  4. Implementation of database design
  5. SQL queries.

- **Course Outcomes.** At the end of the course, the students should be able to:
  1. Extract information from data
  2. Design a database for a given problem; including identifying database attributes, tables and their fields, and table schema
  3. Draw the ERD for a database
  4. Draw the relational schema for a database
  5. Understand the difference between the tables types, i.e. 1NF, 2NF, 3NF, and BCNF
  6. Perform normalization on a database design
  7. Write SQL queries to extract desired information from a database
  8. Use database software such as Microsoft Access to implement a database design and create a functional database application.

COURSE DELIVERY

The main issues in delivering the database to engineering students are the availability of software and computer labs, integrating the software into the course lectures and the use of a project throughout the course. We cover each issue briefly.

Need for database software and computer labs

Many engineering departments have, or have access to, computer labs with Microsoft Office suite, including Access, installed in a network configuration for students to use. While the software and lab availability are not a problem, the logistics for securing and using a computer lab for the course add an extra dimension to administering the course. A lab needs to be reserved early in advance. The number of students allowed to register for the course at a given semester is determined by the number of computer seats available in the lab.

Database software integration into the course

This is always a sticky issue. The central debate is not whether to cover software in the class or not, but when to begin the coverage and how much. We investigated the timing of coverage by testing two hypotheses over two separate semesters. Our hypothesis for the first experiment is 'Students should learn ALL the database concepts before beginning to introduce and use database software'. With this hypothesis in mind, we set to cover database design concepts, tables, redundancy, ERDs, relational schemas, normalization and data dictionary.

As the semester progressed we observed the students' interest and attitude towards the course. At the beginning of the semester, the students were upbeat and excited about the course and the topic. As time went on, they lost interest and were struggling to stay motivated in the computer lab. They also complained directly to the instructor that the material is dry and does not make sense, and that they should use database software early in the course. Students struggled to appreciate the database abstract concepts such as the value of creating relational schema for a database, and the value of establishing the right type of referential integrity between tables. Moreover, they could not understand well the table joins and their types (equijoin, left outer join, and right outer join).

Following the disappointing results of the first hypothesis, our hypothesis for the second experiment was 'Students should learn BOTH the database concepts and use the database software simultaneously'. Our rationale was to use the software whenever possible to illustrate the concepts immediately after introducing them. This approach worked well. Students stayed motivated and attentive throughout the semester. They were engaged with the course and proactive. They would ask questions, and we would answer them theoretically and demonstrate the answer using the software. In retrospect, the success of this approach is attributed to its ability to accommodate different learning styles of students as discussed in the best teaching practices for engineering students such as T4E (Teaching Teachers To Teach Engineers) teaching model and its successors ExCEEd (Excellence in Civil Engineering Education) and ExcEEd (Excellence in Engineering Education) [21–28]. Simply stated, engineering students learn best by doing.

Amount of software coverage

While we agree with the students on using software tools early in the semester, we differ greatly on how much of it we should cover. The use of software requires teaching so students can use it, meaning it takes time away from the course main mission: teaching, not training, the students. Our view on software is that it is a tool and students should acquire just enough skills to use it effectively. The students' view is quite the opposite; they view it as a necessary skill that makes them more marketable when they seek co-op jobs. Once we explain our view to the students, they are convinced.

Class projects

Two interrelated projects were assigned to the class as part of the student learning process and experience. The projects allow the students to practice the concepts they learn in class and apply them first hand. The projects' theme is to design and build a database application. Students are free to select the application as long as it meets the projects' requirements. The application consists of two main parts: front end and back end. The back end is the database itself; its tables and relational schema. The front end is the graphical user interface (GUI) that the database users utilize to store data and/or retrieve information from the database. The students use Access to implement their application.

The first project is the midterm project and concerns the design (using the abstract concepts) and creation (using Access) of the back end of the

application. The timing and the scope of the midterm project are ideal because they coincide with finishing the database design concepts. The second project is the final project and concerns the design (using menus) and creation (using Access) of the front end (GUI) of the application. Students are required to use Access in both projects. By the time the students begin the final project, they know how to create SQL queries and GUIs in Access. Students are able to create SQL queries in Access via either of its two modes: Design View or SQL View. Students may start composing a query in the Design View and finish it by manually editing it in the SQL View. Students are able to do the editing because they understand the syntax of SQL queries.

We guide the students throughout the projects by holding discussions in the class and by providing them with grading rubrics, one for each project. Each rubric has hard and soft requirements. As a sample, Figure 1 is the grading rubric

Six hard requirements (70% of total project grade).	
	Score
1. Business Plan: biz description, biz rules	30
2. Entity Relationship Diagram, complete with relationships and cardinalities	30
3. Relationship Schema (screenshot from Access)	30
4. Normalization. If normalization gives 'strange' results and you would like to override the results (e.g. keep a table in 2NF instead of breaking it to two in 3NF, you must justify and explain in details your decision. The goal is to have a robust DB design.	30
5. Data Dictionary	30
6. Include (minimum): — 4 Tables (both design and data views screenshots) — 2 Queries (design, SQL, and datasheet views screenshots) — One of each: single valued, multi-valued, derived, and composite attributes (describe and explain). If you do not have multivalued type, you must justify and explain in details why not. Also, you cannot implement derived attributes in Access, just explain the attribute and how to calculate it. — One of each: text, number, AutoNumber, Date/Time, Yes/No entity types (include and show in Data dictionary)	50 (15, 15, 10, 10)
Total:	200
DB design & content (15% of total project grade)	
	Score
1. Business Practices are followed and the general database design makes logical sense.	50 pt each
2. Content—the database designed applies to the business description provided, all data is relevant to the project.	
3. Relationships—the relational schema applies all types of relationships (1:1, 1:M, M:N) at least once, all tables are related somehow, well thought out design. If you do not have M:N relationship, you must justify and explain in details why not.	
4. Engaging and interesting The database is expressive and interesting, the business plan is well thought out and applicable.	
Total:	200
Database Theory Questions (15% of total project grade)	
	Score
1. Describe/discuss the expandability/scalability of your DB design. Is your design robust enough to handle future growth? Offer example of an additional piece of data being added to a table.	50 pt each
2. Pick one of your DB relationships, describe/discuss how the relationship prevents bad data and deletion of records	
3. Did you make any fields that are required 'not nullable'? If yes, why did you do so, if not why not? Discuss and be specific to your DB. Note: Access provides 'Required' property in table design view.	
4. What did you learn from the project, how much time did you spend on it? How many revisions did your DB take before achieving your objectives? Discuss	
Total:	200

Fig. 1

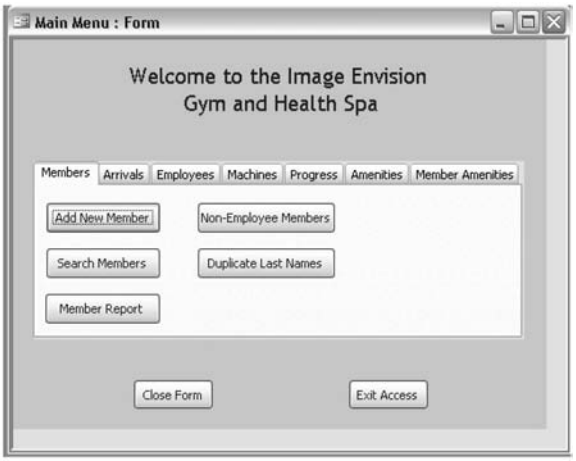


Fig. 2. Main menu of Gym and Health Spa Application.

for the midterm project: 'This rubric shows what you are expected to complete for the midterm project, and how it will be graded.'

Sample projects

Students do a great job on their projects because they own them; students work in groups of two and select their own projects. Students are given the choice of working individually. However, we explain to them the pros and cons of team work and encourage them to work in groups as this mimics professional life after they graduate. Here are sample projects from the last course:

1. Gym and Health Spa Application—this allows the gym staff to track arrival times of the gym members in order to determine the busiest times of the day/week—a graph of the gym traffic vs. time of day is posted for the convenience of members. As an additional service, members have the option to track their workout progress in the gym. Figures 2 and 3 show the application.
2. Library Loan Application—this creates a database to allow a library to track book loans and returns effectively. The database replaces the current card catalog and enables a library to expand its services to town residents. Figures 4 and 5 show the back end of the application. Figure 6 shows a user form to search for a book.
3. A Bank Application—allows a bank to issue accounts and loans to its customers. The bank does not require a customer to open an account to obtain a loan. A customer may have multiple accounts. Figure 7 shown an SQL query and its results.

PEDAGOGICAL APPROACH

We utilize the Productive Pedagogy (PP) approach [29, 30] as a framework to deliver the highest intellectual quality for student learning. We apply the four elements of PP throughout the semester as follows:

1. Intellectual quality. We engage the students in complex understanding as opposed to 'spoon feeding' them. We guide them into higher-order thinking, deep knowledge, deep understanding and clear idea communication. We provide them with core knowledge in the classroom and encourage them to investigate further outside the classroom. One student wrote: 'We really learned how to do it [create Access forms, macros and reports] on our own; and that was actually very gratifying!' We design the homework problems and exams to help the students develop their critical thinking about databases. One student wrote: 'I feel as though the homework assignments were, for the most part, productive and thought provoking. They aided us in developing our database design

Employee ID	Last Name	First Name	Hire Date	Position	Pay
100	Baker	Barry	15-Dec-00	Manager	$45,000.00
101	Dil	Paula	17-Dec-00	Trainer	$32,000.00
102	Julep	Jerry	17-Dec-00	Maintenance	$26,000.00
103	Russo	Martin	02-Jan-01	Front Desk	$23,000.00
104	Blume	Betsy	04-Mar-02	Trainer	$29,000.00
105	Asay	Stephen	01-Dec-07	Business Manager	$75,000.00
106	Baker	Bonnie	01-Dec-07	Secretary	$30,000.00
107	Hale	Lauren	30-Nov-07	Facility Planner	$70,000.00
108	Mueller	Elsa	01-Nov-07	Trainer	$35,000.00
109	Eng	Kim	01-Nov-07	Trainer	$35,000.00
110	Dean	Missy	01-Jan-04	Front Desk	$24,000.00

Fig. 3. Employees information report.

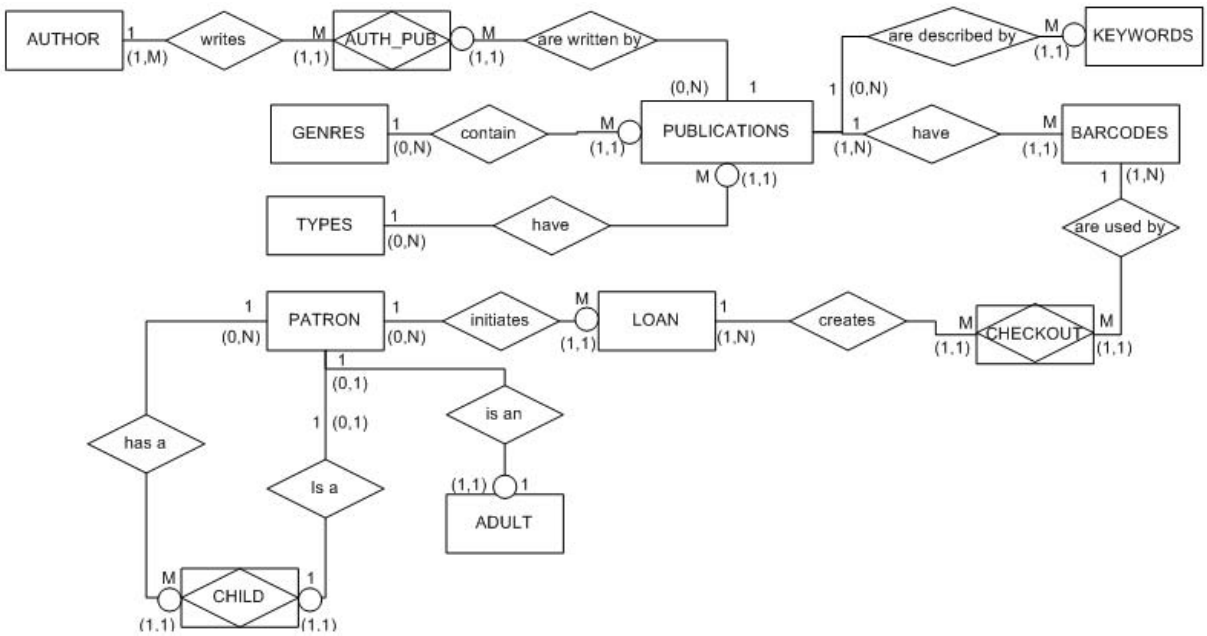


Fig. 4. Library ERD.

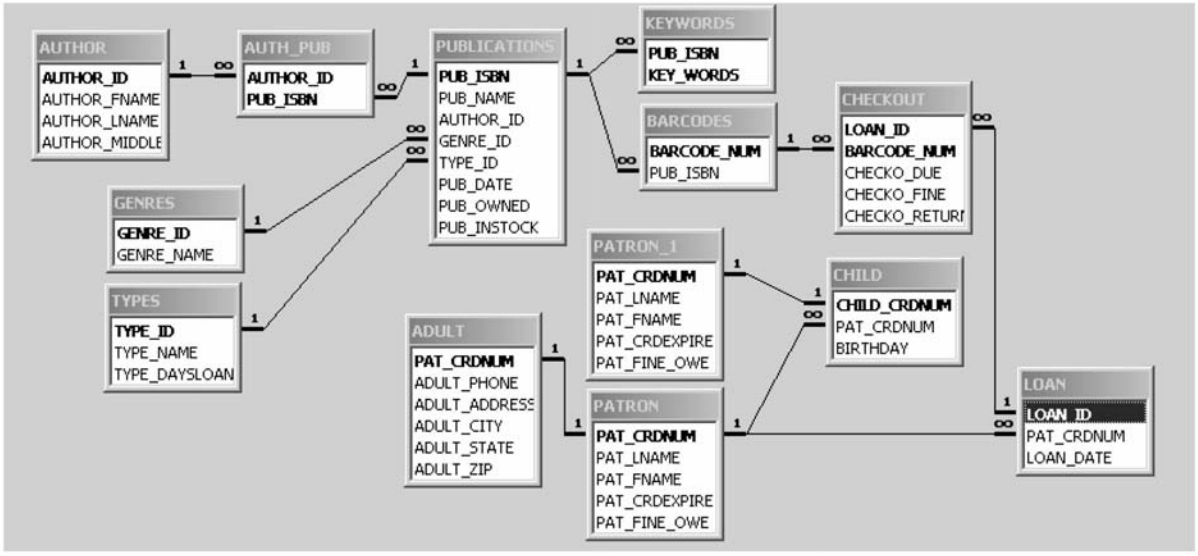
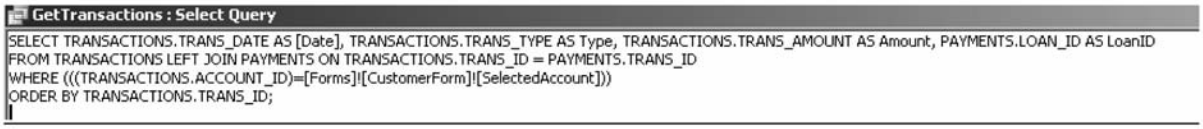
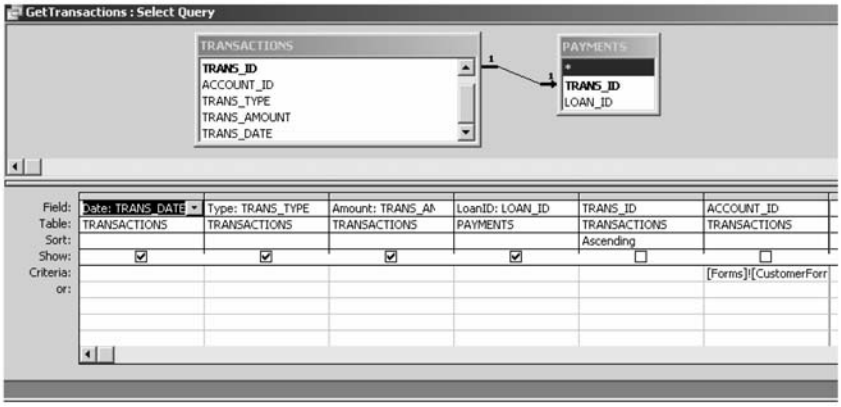


Fig. 5. Library Relational Schema.

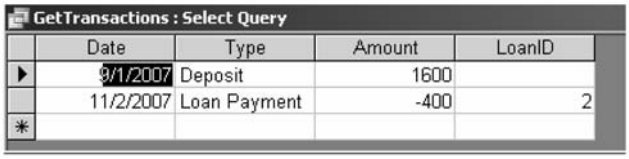
Fig. 6. Form to search for a library book.



(a) SQL View



(b) Design View



(c) Query Results

Fig. 7. A left outer join SQL query.

skills from nothing into skills that can at the very least build a meaningful and functional database application'.

2. Relevance and connectedness. This element means helping the students to value what they learn and relate and connect the classroom material to their past experiences and the real world. We, the instructors, always discuss database problems from consulting activities or case studies that we can find. Also, we invite and encourage students from the class who have had database related co-op experience and jobs to share them with the students in the class. One student wrote about the course: 'I knew NOTHING about databases coming in . . . and could design one to be used on co-op.' Another wrote: 'We feel that as we might have thought that creating the ERD was stupid and pointless, now we see the value it added.'
3. Socially supportive learning environment. Here, we involve the students to influence the class activities and how they are implemented. We explain the issues to the students, give them choices and ask them to vote on what to do. A case in point: we needed to decide which software option to select—Microsoft Access or MySQL and PHP. Students chose the former as we discussed earlier in the paper. In this regard, two students wrote: 'We're glad the class took a turn to learning SQL/Access because we feel that this knowledge will allow us to expand our job capabilities both on our last co-op and in industry in a few years.' Another social aspect includes encouraging peer-to-peer learning inside and outside the classroom. We encourage students to brainstorm together on course activities, but yet work independently on writing and documenting their work in preparation for future exams. We also encourage students to work in groups of two for the course projects. One student wrote about working in groups: 'In total, we spent approximately 10 hours on this final project . . . so if the project had been done by only one of us it would have likely taken in excess of 15 hours.'
4. Recognition of difference. We interpret difference here as between students in cognition and understanding speed of material covered in the class. Difference also encompasses the different styles of student learning; some students understand better at an abstract level, while other are visual and like hands-on activities. With this in mind, we blend all teaching approaches from abstract, to visual, to hands on in such a way to meet all student learning styles, needs and cognition needs. One student wrote: 'I had a lot of fun! This is a very useful skill and I think I will definitely use it in the future.' Another wrote: 'I feel like I learn a lot of things from Database by using Access. The text book is very helpful, and the TA is very helpful. The pro-

fessor writes all the detail on the board during lecture which is good for us.'

While most students were satisfied, some voiced their concerns (three out of 38). One student wrote: 'We found it frustrating to try to apply a certain type of query to our database where it was clearly not needed, and would never be practical to implement'. Another wrote: 'Stress the join query a little bit more'. And a third: 'Spend less time in the course talking about database design and theory and more time actually working with Access'.

PERFORMANCE CRITERIA

The performance criteria are the specific measurable activities required to confirm that we have achieved the course outcomes. We listed these at the beginning of this paper. The corresponding performance criteria (one-to-one mapping) to achieve these outcomes are:

1. Find information in a given problem.
2. Create table schema, fields and records to store given data.
3. Sketch ERD using a tool such as Word or Visio.
4. Sketch the relational schema between tables of a database using database software such as Microsoft Access.
5. Evaluate the tables of a database and classify them based on their NF type.
6. Convert all tables of a database to the preferred 3NF type.
7. Use the SELECT and action queries of SQL.
8. Use Microsoft Access to evaluate different database designs and to implement the class projects.

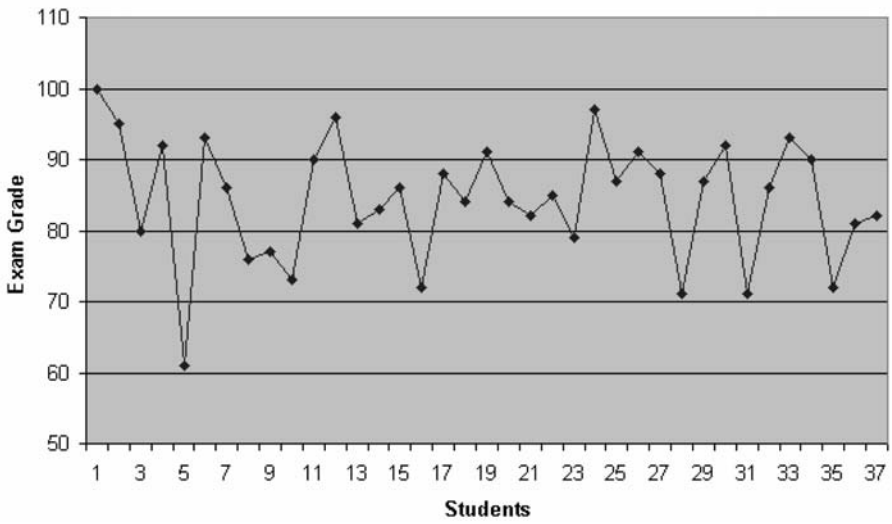


Fig. 8. Class grade distribution for an exam.

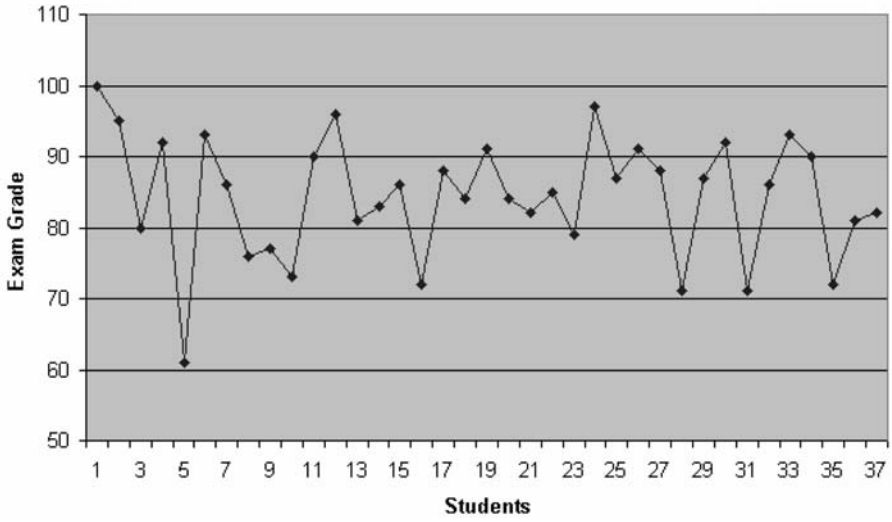


Fig. 9. Class Final grade distribution.

ASSESSMENT

We assess the students' performance in the course via both formal and informal assessment tools. Formal tools include homework, exams and projects. The percentage split is 20 per cent homework, 50 per cent exams, and 30 per cent projects. We give nine sets of homework that we grade and return to the student with constructive feedback. We average about one homework per week. Each weekly homework corresponds to the weekly material coverage. We give two exams during the semester. These exams are closed book and test the abstract concepts of the course. Students are allowed one page of notes for each exam to eliminate the need for memorization; we emphasize critical thinking in the exams. As for the projects, we assign two separate, but related, projects as we discussed earlier in the paper. Each project is worth 15 per cent of the total course grade. Each project is graded based on a rubric published at the time of assigning the project, so students know what is expected of them. We have included the midterm project rubric in the paper as a sample. All students' work is graded and returned to them in a timely fashion to provide feedback before the next assignment. Informal assessment tools include class discussions and in-class hands-on activities.

In addition to assessing the students, we also like to assess the course itself. Formal course assessment includes the students' evaluation mandated by the university at the end of the semester. Informal course evaluation includes asking students occasionally to provide their candid feedback about the course and encourage them to recommend changes in the teaching style to meet their learning needs.

EVALUATION

We make a conscious effort to use all the assessment tools at our disposal to continually evaluate student performance and course progress during the semester. We listen to the students' complaints about the course load and make adjustments if needed. We adjust the course speed as well. Students seem to grasp the material well and seem motivated and interested as evidenced by their attendance and grades. Attendance is always high in the class. Grades are always very good, e.g. average grade in exams is between 80 and 85 out of 100. Figures 8 and 9 show the class grade distributions for an exam and the final grade respectively.

CONCLUSION

The paper presents an effective learning model to teach databases and other IT subjects to engineering students. The model is based on the right balance of theory and practice to deliver the course material in a way to make students understand and apply what they are learning, and at the same time enjoy their learning experience. One student wrote: 'Databases are much more complicated than we previously thought'. Another wrote: 'Thanks to this project, we now have the confidence to design an entire database and have even considered using our newly honed skills for personal/professional project in the future'. Yet, another wrote: 'That concept [join query] is so important to DB design because it answers why we must use relationships (the good vs. bad design). Overall, great job. I had fun.' Other students wrote: 'Both of our experiences during class and homework time was beneficial for our learning and overall, a pleasant experience!'

REFERENCES

1. *The Engineer of 2020: Visions of Engineering in the New Century*, National Academy of Engineering, available at http://www.nap.edu/books/0309091624/html/ (2004).
2. *Educating the Engineer of 2020: Adapting Engineering Education to the New Century*, National Academy of Engineering. Available http://books.nap.edu/catalog/11338.html (phase I) and http://www.nap.edu/books/0309096499/html/ (phase II), (2005).
3. J. H. McMasters, B.J. White and T.H. Okiishi, *The Industry-University-Government Roundtable for Enhancing Engineering Education*, AIAA 99-0281, presented at the 37th AIAA Aerospace Sciences Meeting, Reno, NV, January, (1999).
4. B. Kayis, *Trends in Undergraduate Mechanical, Manufacturing, Aerospace and Mechatronics Programs Worldwide: In-depth Study of 55 Universities' Programs*, Proceedings—IEEE International Engineering Management Conference: Innovation and Entrepreneurship for Sustainable Development, IEMC (2004), pp. 178–182.
5. J. H. McMasters, Influencing Engineering Education: One (Aerospace) Industry Perspective, *Int. J. Eng. Educ.*, **20**(3), 2004, pp. 353–371.
6. R. B. Freeman, *Does Globalization of the Scientific/Engineering Workforce Threaten U.S. Economic Leadership?,* National Bureau of Economic Research (NBER), NBER Working Paper No. W11457, (2005).
7. Erik De Graaff and W. Ravesteijn, Training complete engineers: global enterprise and engineering education, *Eur. J. Eng. Educ.* **26**(4), 2001, pp. 419–427.
8. J. Yeargan and K. Hernaut, The globalization of European engineering education: an American observer's perspective, *IEEE Frontiers in Education Conference*, **3**, 2001, pp. 1–5.

9. K. Sheppard, Preparing Engineering Students for the New Business Paradigm of International Teamwork and Global Orientation, *Int. J. Eng. Educ.*, **20**(3), 2004, pp. 475–483.
10. F. Bullen, The Broad and Strategic Value of the Freshmen Engineering Experience—FEE, *Int. J. Eng. Educ.*, **22**(6), 2006, pp. 1241–1251.
11. E. J. Coyle, L. H. Jamieson and W. C. Oakes, EPICS: Engineering Projects in Community Service, *Int. J. Eng. Educ.*, **21(**1), 2005, pp. 139–150.
12. L. Neal, K. Kratchanov and I. Stanev, Developing a Western-Style Information Systems Engineering Curriculum and Infrastructure, *Int. J. Eng. Educ*. **14**(1), 1998, pp. 48–53.
13. http://www.microsoft.com/presspass/press/2005/nov05/11-01PreviewSoftwareBasedPR.mspx
14. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 4th Ed., Addison-Wesley, (2004).
15. C. J. Date, *An Introduction to Database Systems*, 8th Ed., Addison-Wesley, (2004).
16. P. Rob and E. Semaan, *Databases*, 2nd Ed., McGraw-Hill, (2004).
17. R. Ramakrishnan and J. Gehrke, *Database Management Systems*, 3rd Ed., McGraw-Hill, (2003).
18. A. Silberschatz, H. F. Korth and S. Sudarshan, *Database System Concepts*, 5th Ed., McGraw-Hill, (2006).
19. J. A. Hoffer, M. B. Prescott and F. R. McFadden, *Modern Database Management*, 7th Ed., Prentice-Hall, (2005).
20. J. A. Hoffer, M. B. Prescott and F. R. McFadden, *Modern Database Management*, 8th Ed., Prentice-Hall, (2007).
21. C. H. Conley, S. J. Ressler, T. A. Lenox and J. W. Samples, Teaching Teachers to Teach Engineering, *J. Eng. Educ.*, **89**(1), 2000, pp. 31–38.
22. N. D. Dennis, ExCEEd Teaching Workshop: *Taking It on the Road*, 2001 ASEE Conference & Exposition: Peppers, Papers, Pueblos, and Professors; Albuquerque, NM, (2001), pp. 24–27.
23. R. Welch and A. Estes, Teaching Pedagogy 101, 2005 ASEE Conference & Exposition: *The Changing Landscape of Engineering and Technology Education in a Global World,* Portland, OR, (2005), pp. 20–23.
24. R. Felder, Matters of Style, *ASEE Prism*, **6**(4), 1996, pp. 18–23.
25. P. Wankat and F. Oreovicz, *Teaching Engineering*, McGraw-Hill, (1993), pp. 244–305.
26. J. Lowman, *Mastering the Techniques of Teaching*, Jossey-Bass, San Francisco, CA, (1995), pp. 20–31.
27. D. H. Jonassen, *Computers in the Classroom: Mind Tools for Critical Thinking*, Prentice-Hall, (1998), pp. 20–30.
28. Project-Based Learning: Building Communities of Reflective Practitioners, *Management Learning*, **32**(1), 2001, pp. 61–76.
29. J. Allan, Productive pedagogies and the challenge of inclusion, *Brit. J. Special Education*, **30(**4), 2003, pp. 175–179.
30. F. M. Newmann, H. M. Marks and A. Gamoran, Authentic Pedagogy and Student Performance, *Amer. J. Educ*. **104**(4), 1996, pp. 280–312.

Abe Zeid is a Professor of Mechanical and Industrial Engineering and Director of the CAD/CAM Program at Northeastern University, Boston, MA. He has published extensively and has written highly acclaimed textbooks in the areas of CAD/CAM and Computer Information Systems. His research interests are in CAD/CAM and real-time data processing. He teaches IT and CAD/CAM courses including databases and computer information systems.

Sagar Kamarthi is an Associate Professor of Industrial Engineering and Associate Director of the Laboratory for Responsible Manufacturing at Northeastern University, Boston, MA. He received his MS and Ph.D. degrees from the Pennsylvania State University. His research interests are sensor-embedded products, advanced sensor data processing and interpretation and neural networks. He teaches courses in neural networks and expert systems.