BLOG @CACM

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3014349

<http://cacm.acm.org/blogs/blog-cacm>

How We Teach CS2All, and What to Do About Database Decay

Valerie Barr considers how to make computer science education meaningful and relevant to all, while a team from the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory offers strategies to counter database decay.

Valerie Barr
How We Teach Should Be Independent of Who We Are Teaching

<http://bit.ly/2eYnx0Z>

October 11, 2016

For many years I have been part of discussions about how to diversify computing, particularly about how we recruit and retain a more diverse cohort of computer science (CS) students. I wholeheartedly support this goal, and spend a considerable amount of my effort as chair of ACM-W helping to drive programs that focus on one aspect of this diversification, namely encouraging women students to stay in computing.

Of late I have become very concerned about how some elements of the diversity argument are being expressed and then implemented in teaching practices. A shorthand has developed that often comes out as two problematic claims:

Problem 1. Women are motivated by social relevance, so when we teach

them we have to discuss ways in which computing can contribute to the social good.

Problem 2. Students from underrepresented minorities (URM) respond to culturally relevant examples, so when we teach them we have to incorporate these examples into course content.

This formulation of what we should be doing in the classroom is problematic for a number of reasons:

1. These statements are silent on the subject of white and Asian men, the groups that dominate in CS classrooms, effectively implying that these people are not interested in computing for the social good or culturally relevant examples, that they are only motivated by the hard-core geeky techie parts of computing.

2. This formulation paints all women with a single brush, and does the same for URM students. Some women are interested in the social relevance of computing, but are all women going to be motivated by this? Some URM

students are motivated by culturally relevant examples, but are all URM students going to be motivated by this?

3. While painting women and URM with a single brush, this formulation implies that members of these groups are not interested in computing for techie reasons, that members of these groups will not ever be excited about the technology in its own right.

4. Further, there is an implication that we need to discuss the social relevance of computing *only* when there are women in the class, and we need to utilize culturally relevant examples *only* when there are URM students in the class.

5. The logical, and dangerous, final conclusion is that if there are *only* Asian and white men in the room, then we do not need to make any changes at all to course content or pedagogy.

These assumptions about students can have a very negative impact on our teaching, causing us to potentially drive away the very students we are hoping to recruit and retain. As we continue efforts to diversify computing, we cannot afford to paint *any* group in a monochromatic way. We have to embrace the richness of today's student population by making what we teach meaningful and relevant to them. There are women who want to geek out about hard-core tech, and there are men who care deeply about computing for the social good. There are students of all genders and ethnic and racial backgrounds who will be happy with an old-fashioned lecture, and those who will thrive on

active learning with examples drawn from a range of cultures and application areas. Many students will be motivated by knowing how the techniques and subject matter they are learning fit into their future workplace or life goals.

In order to change the toxic climate in tech, a climate that, for example, leads 45% of women to leave tech jobs within five years, we have to teach *everybody* differently. If we pretend that all women students are the same, and all URM students are the same, and all Asian and white male students are the same, then we will never adequately address the blind spots and weaknesses in instruction and curriculum development that have led to our current situation. A rich approach to curriculum and teaching pedagogy will maximize our ability to reach *all* kinds of learners, all parts of the student population. We have to use varied content and pedagogies regardless of whom we see in the room and work to connect to what students know or care about. This approach will guarantee that all students, including those from the groups that currently dominate computing, will be exposed to a rich, multifaceted, view of computing, be better equipped to address the challenges of the field, and be better equipped to work collegially within a diverse workforce.

Thanks to several colleagues who gave me important feedback on prior versions of this post.

**Michael Stonebraker,
Raul Castro Fernandez,
Dong Deng, and
Michael Brodie**
**Database Decay and
What To Do About It**

<http://bit.ly/2eDQArS>

October 24, 2016

The traditional wisdom for designing database schemas is to use a design tool (typically based on a UML (https://en.wikipedia.org/wiki/Unified_Modeling_Language) or ER (https://en.wikipedia.org/wiki/Entity-relationship_model) model) to construct an initial data model for one's data. When one is satisfied with the result, the tool will automatically construct a collection of 3rd normal form relations for the model. Then, applications are coded against this rela-

tional schema. When business circumstances change (as they do frequently), one should run the tool again to produce a new data model and a new resulting collection of tables. The new schema is populated from the old schema, and the applications are altered to work on the new schema, using relational views whenever possible to ease the migration. In this way, the database remains in 3rd normal form, which represents a "good" schema, as defined by DBMS researchers. "In the wild," schemas often change once a quarter or more often, and the traditional wisdom is to repeat the above exercise for each alteration.

In a survey of 20 database administrators (DBAs) at three large companies in the Boston area, we found that this traditional wisdom is rarely-to-never followed for large, multidepartment applications. Instead, DBAs try very hard not to change the schema when business conditions change, preferring to "make things work" without schema changes. If they must change the schema, they work directly from the relational tables in place. Using these tactics, the ER or UML model (if it ever existed) diverges quickly from reality. Moreover, over time, the actual semantics of the data tend to drift farther and farther from a 3rd normal form data model.

We term this divergence of reality from 3rd normal form principles **database decay**. Over time, decay becomes worse and worse, leading to **rotted** databases and ultimately to databases that are so decayed that they cannot be further modified. Obviously, this is a very undesirable state of affairs.

In our opinion, the reason for decay stems from the multidepartment organization of large implementations. Hence, various pieces of the overall application are coded by different organizations, typically using ODBC (https://en.wikipedia.org/wiki/Open_Database_Connectivity) or JDBC (https://en.wikipedia.org/wiki/Java_Database_Connectivity) to specify the SQL in transactions. If one business unit needs to change the semantics of the database, it is exceedingly difficult to figure out what code from other departments needs to be changed and how extensive the required repairs are. In our opinion, this leads DBAs to change the schema in such a way that application maintenance is minimized,

rather than making a change that maximizes the cleanliness of the data. Of course, the result of a different DBA cost function is database decay and rot.

Seemingly, database decay is a fact of life, which ultimately renders databases unable to be modified. There are three strategies that can counter database decay.

The first one is to construct **defensive schemas** in the first place. Such schemas are more resilient to subsequent changes than ones produced using the traditional wisdom. We have developed a methodology for such schemas, which will be addressed in an upcoming paper.

The second tactic is to write **defensive application code**. For example, one should never use `Select * From Table-name`, because it tends to make applications break if attributes are added or deleted downstream.

Lastly, in our opinion, it is a bad practice to let application groups directly code against an ODBC/JDBC interface. This is what is responsible for DBAs not knowing the impact of possible schema changes. Instead, we advocate requiring application groups to use a messaging interface to send higher-level commands to a DBMS. These messages are intercepted and turned into SQL in server-side code. Such an architecture localizes DBMS code that may need to be changed later on. Moreover, we have written a prototype system that can examine such code and determine if it needs to be changed as a result of schema evolution. In this way, we expect to lower the cost of schema changes, and perhaps slow down or arrest database decay. An upcoming paper details our prototype.

We are looking for "in the wild" database projects that are dealing with schema evolution that would be amenable to trying our prototype system. If you are interested, please contact Michael Brodie at mlbrodie@mit.edu.

Valerie Barr is a professor in the Computer Science department of Union College, and serves as chair of ACM-W, the ACM Council on Women in Computing.

Michael Stonebraker is an adjunct professor in the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory (CSAIL), and recipient of the 2014 ACM A.M. Turing Award.

Michael Brodie is a Research Affiliate in CSAIL, while Dong Deng and Raul Castro Fernandez are post-doctoral researchers at CSAIL.

© 2017 ACM 0001-0782/17/1 \$15.00