This is the sixth in a series of articles on database management and security. In this article, we look at data anomalies that can result from poorly-structured database designs and how the process of normalization helps avoid such problems.

Before Dr E. F. Codd and his colleagues formalized the reasoning for deciding how to group data into records (relations, rows, tuples), programmers were always running into problems that qualify as security issues: they would discover that users were causing their files or primitive databases to lose critically important data.

Suppose we naïvely defined an inventory record that put all the details about items in a store into a single record? How would we keep track of the items we in our store if we delete the last record that includes the item information? How do we store information about an item that nobody has bought yet? Why are we storing copies of the information about the same items and the same people?

28. This slide gives a couple more examples of the deletion anomaly that results from badly normalized data structures. You can see that these erroneous record layouts mix information about fundamentally different entities. There’s no good reason to store details about a doctor in the record pertaining to a patient. There’s no good reason to store information about a car in the same record as price information about types of repairs. These non-normalized data structures don’t make sense even if we don’t know much about databases.

29. Just as we saw in previous slides, a dataset that mixes detailed information about a part with radically different information about where the parts are stored causes a real headache. We need to separate the information about the parts from the information about how many there are and where they are kept. What we can do is to keep a record that links a specific part number to a specific bin number and lists how many of that part are in that bin. Now that is a normalized design and it easily handles information about parts that are out of stock, bins that are empty, and parts that are in more than one bin.

30. We’ve already seen some of the problems that can occur in a poorly structured database when we delete records. Here are some more. In particular, I draw your attention to what could happen if we have a library database in which we delete the record with information about the publisher when there are many books that point to that publisher. You can see that removing the only record that tells us the name, address, telephone number and so on for the publisher could be very harmful for the library and would make the book records that point to the nonexistent publisher record meaningless (for example, they might have the number 2345 which used to point to Random House which now wouldn’t mean anything at all).

Databases have strict rules which make it impossible to cause this kind of havoc; we call these rules referential integrity constraints. Other examples include not being able to add a record that duplicates a unique key value in a dataset or not being allowed to add a record that points to a nonexistent key. As an example of the latter, an order-entry system would prevent an operator...
from adding an order for a client that does not yet exist in the database. First you add the client record, then you can add the order placed by that client.
You are familiar with such constraints simply from having bought things on the Web, where it is perfectly normal for us all to fill out an identification form about ourselves before we can place our order containing the items we want to buy. The next time, however, if we allow the vendor to keep our information, we can just identify ourselves and fill out the details of the new order.

* * *

M. E. Kabay, PhD, CISSP-ISSMP <mailto:mekabay@gmail.com> specializes in security and operations management consulting services. CV online.<http://www.mekabay.com/cv/>

Copyright © 2009 M. E. Kabay. All rights reserved.

Permission is hereby granted to Network World to distribute this article at will, to post it without limit on any Web site, and to republish it in any way they see fit.