Ideally, a record's keyfield should be little more than a meaningless serial number.

change ID numbers on file, the lion's share of our records (the noncomputerized part) will be wrong.

Visualize the consequences by imagining a user who's been monitoring a ship's progress across the ocean. One day, inquiring about it with the old ID number, she's told that no such shipment exists. Put yourself in her shoes: she knows it was out there yesterday, and she knows it didn't arrive anywhere today. Have you ever wondered how that business about the Bermuda Triangle got started? Now you know.

Notice, by the way, that if we bring up this issue during design, the user will inevitably say, "Oh, but those data [whatever they are] will never change!" Now, if you believe this, get in touch with me—I have a Caribbean island I'd like to sell you.

The solution is to keep data out of the keyfield. Fields in database records come in two flavors: identification and description. ID numbers identify; data fields carry data. The keyfield identifies the entity that the record models (a shipment), while all the other fields describe its attributes. Put origin, destination, cargo, etc., in the body of the record. If users want output sorted by these fields, do it. If they want to retrieve on-line using them, let them. But keep them out of the keyfield. Ideally, a record's keyfield should be little more than a meaningless serial number.

THEY DON'T TRUST US

Why do some users insist on embedding data into keyfields? It's because they don't trust us. Remember your last payables system? The vendor master was up and used by hundreds of programs when the user wanted to add "vendor industry type" to it. Since the record's filler had been consumed years before, you realized you'd have to make the record longer. To do this, you'd have had to track down, modify, and recompile all those undocumented programs. So you told him that it would take six months and cost tens of thousands of dollars. He withdrew the service request, but he still needed the data, so he simply assigned blocks of ID numbers to the different industry types. As the years went by, he got into the habit of structuring ID numbers with data and has never stopped.

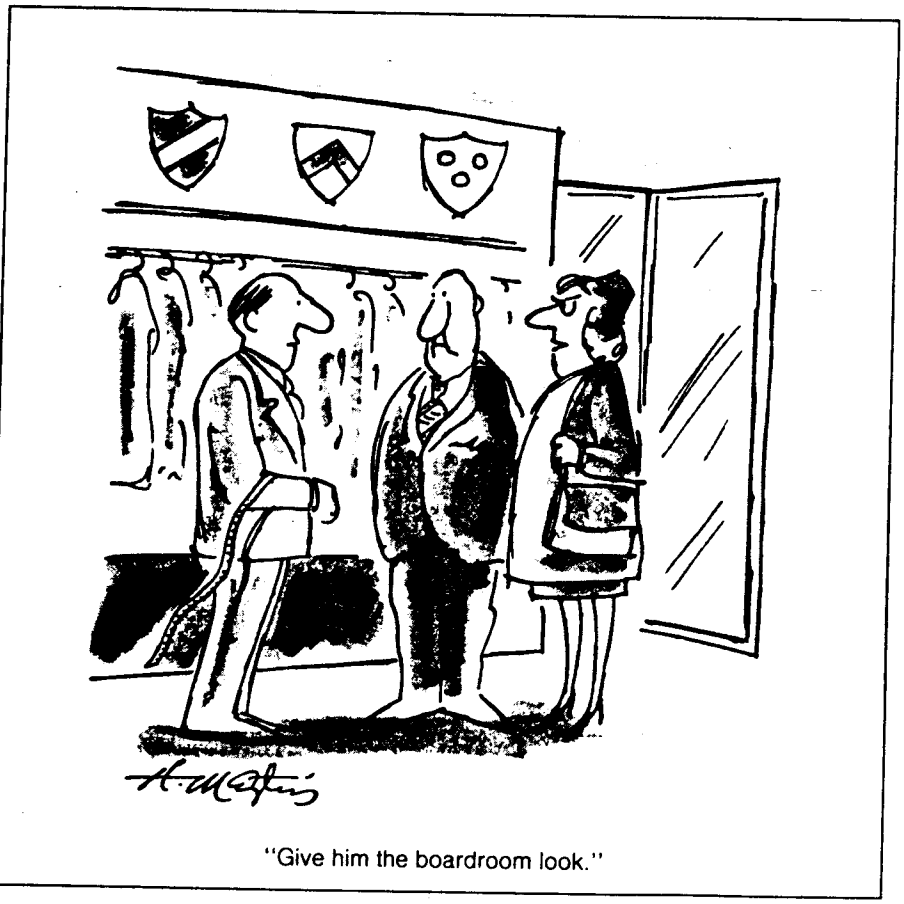
The solution centers on our ability to react quickly to the changing business environment, adding new items that can be used as secondary access keys as needed. With today's tools, we can provide this level of service. But modern tools have their own drawbacks. Each keyfield, besides being dataless and unchanging, should be unambiguous. In other words, you should not have different real world entities (two employees, for example) on file with the same ID number.

The problem emerges when we misapply technology. Older data access methods (ISAM, VSAM) made it difficult or impossible to write multiple direct-access records with the same key. Today's database packages allow it, but it's still unwise.

One way the problem arises is through shortsighted choice of ID number. Case history: the Personnel File. Everyone agreed to use social security number as employee ID number in the new system. The application was installed and turned over to the client before someone asked what to do about Brazilian employees (they're the ones who ship all that coffee). Brazil is a sovereign nation and its citizens aren't issued U.S. social security numbers. The proposed solution was to invent a number (999-99-9999) and use it for every Brazilian. The user's question was, "You mean the database won't let me have two employees with the same ID number?" Our too-truthful answer was that the package would, in fact, allow such an aberration. Needless to say, the file is now a bit tricky to update. It also produces interesting telephone conversations between personnel managers: "Not that 999-99-9999, Harry, the other 999-99-9999!" (accompanied by a great deal of arm waving).

Finally, keyfields should be unique. You shouldn't have two records (with different ID numbers) for the same vendor. The problem arises when each of two different user organizations wants total authority over the same file. They compromise by splitting the range of possible ID numbers between them. Each maintains the records within his or her range of ID numbers. But since nobody coordinates new vendor numbering with anyone else, common vendors get two numbers. The results are interesting but not very safe. Several years ago, my then-employer got stuck with a $150,000 bad debt from a deadbeat corporation. The offender's record was immediately tagged as "NO CREDIT-BAD RISK" by our credit department. Three weeks later, they stuck us again for $200,000. Same outfit, different corporate ID.

Next time, we'll examine "The Two-Headed Arrow." ◉



"Give him the boardroom look."

Frank Sweet is corporate manager of data administration for the Charter Co., Jacksonville, Fla.