DOI:10.1145/2447976.2447980

Try Old Boys Security Network

[redacted]

[redacted]

Relational Model Alive and Well, Thank You

Carl Hewitt's letter to the editor "Relational Model Obsolete" (Jan. 2013) betrayed a shallow and too-common level of understanding of the relational model. Of the five specific claims he made regarding "limitations" of the model, none is valid:

Inexpressiveness. This was simply wrong. Negation and disjunction are easily in fact, almost trivially expressible. Type generalization and specialization are easily expressible, too, though, to some extent, this is more an issue for the accompanying type system than for the relational model as such.

Inconsistency non robustness. This one was both wrong and confused. Suffice it to say that "*p* AND NOT *p*" is an inconsistency, but "Alice says *p* AND Bob says NOT *p*" is certainly not. Moreover, even if a database really does contain an inconsistency, the relational model would still function (so we are not talking about a problem with the model); rather, the problem is with the database and with the consequent fact one cannot trust the answers given by the system. Further, a query language based on logic that encourages logical contradictions is nonsense.

Information loss. Whether one's updates "lose information" is entirely up to how one uses the model; it has nothing to do with the relational model. One of us (Date) co-authored a book[1] 100% devoted to the use of the relational model to manage temporal data and thereby not "lose information." For the record, the relational model requires no "correction," no "extension," and, above all, no perversion, for this desirable aim to be realized.

Lack of provenance. This point has to do not with the model as such but with how it is used. Note "Alice says" and "Bob says" are provenance information. In fact, the relational model is ideally suited to recording such information, and even SQL DBMSs are widely used for this purpose.

Inadequate performance and modularity. Criticizing the relational model for having no concurrency abstraction is like criticizing a cat for not being a dog. (Hewitt said it is SQL that has no concurrency abstraction, but SQL and

the relational model are not the same thing; indeed, SQL has little to do with the relational model.) As for "a... type should be an interface that does not name its implementations," and to the extent we even understand this remark, types in the relational model meet this criterion.

We would never publish a critique of (for example) Hewitt's actor model without understanding it well; why then does he feel he can publish a critique of the relational model, when he demonstrably does not understand it?

C.J. Date, Healdsburg, CA, and
D. McGoveran, Deerfield Beach, FL

Reference
1. Date, C.J., Darwen, H., and Lorentzos, N A. *Temporal Data and the Relational Model.* Morgan Kaufmann, San Francisco, 2003.

Relational Model Outgrown

Unfortunately, Date and McGoveran make no good arguments against the limitations of the relational model, as outlined in my letter (Jan. 2013), partly because we are using incommensurable terminology (such as "negation," "disjunction," "concurrency," and "abstraction"); for details, see articles in *Proceedings of Inconsistency Robustness 2011* (http://robust11.org), especially regarding the requirement for inconsistency robustness. My point is not to dismiss relational databases as a failure but to build on their success and overcome their limitations.

Such an effort must meet several requirements:

Compatibility. All capabilities of current relational databases must continue to be implemented; in addition, incrementally integrating inconsistent information from multiple relational databases with incompatible schemas (something awkward in the relational model) must become standard practice; and

Natural language + gestures (as lingua franca for communication with computer systems). Semantics of natural language and coordinated gestures of multiple participants must be expressible. An important consequence is expressions and gestures must be able to mutually refer to each other, something awkward in the relational model; for example, if Alice points her finger at Bob and charges, "I accuse you of harassing me," and Bob retorts, "I deny it!," then the mutual co-reference of language and gesture must be expressible.

To move beyond the relational model, I propose the actor model because it already meets these requirements, in addition to the ones I included in my earlier letter. I further propose ActorScript[1] as a more appropriate foundation than SQL for a family of languages for information integration, as it includes the following characteristics:

Security and safety. Applications cannot directly interfere with one another.

Excellent support for concurrency.

- Not restricted to just the transactional model of concurrency, as in SQL;
- Messages directly communicated without requiring indirection through channels, mailboxes, pipes, ports, or queues;
- Integration of functional, imperative, logic, and concurrent programming; and
- Low-level mechanisms (such as threads, tasks, locks, and cores) not exposed by programs.

Language extension. ActorScript has excellent meta-language capabilities for implementing extensions without interfering with existing programs.

Capabilities for extreme performance.

- No overhead imposed on implementation of actor systems; for example, message passing has essentially the same overhead as procedure calling and looping; and
- Concurrency dynamically adapted to available resources and current load.

Relational databases have been an outstanding success and become dominant in the commercial world. However, computing has changed dramatically in the decades since the relational model was developed. Consequently, the relational model and SQL have become obsolete due to the limitations I've outlined here, and innovations like the actor model and ActorScript are required to address current and future needs.

Carl Hewitt, Palo Alto, CA

Reference
1. Hewitt, C. *Tutorial for ActorScript.* arXiv. Cornell University, Ithaca, NY, Mar. 2013; http://arxiv.org/abs/1008.2748

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.