

Class 36: SQL (2): SEQUEL

Held: Friday, April 27, 2007

Summary: Today we trace the evolution of Codd's ideas into a usable programming language.

Related Pages:

- EBoard.
- Reading: Chamberlin and Boyce - SEQUEL: A Structured English Query Language.

Notes:

- EC for alumna scholar talk by Dr. Nesbitt on Monday.
- Presentations next week still require reading responses (and therefore readings).
- I need Wednesday's reading ASAP.

Overview:

- Codd's Contributions.
- Why Read This Paper?
- Thinking About Relations.
- Operations on Relations.
- Relations as Programming Language.
- Describing a Language.
- What's Missing from SEQUEL?

Codd's Contributions

What are the primary contributions of this paper? Here are some.

- A clear attempt to separate what you want to do with data from how the data are represented.
- A model of data that persists to today.
- The use of formal systems to explain the model.

Why do we care about separating access from representation?

- Easier to modify the representation.
- Easier to write programs.
- Easier to modify the representation *on the fly*, as you observe (or are warned of) different patterns of access.

Why Read This Paper?

Up to you to answer this question.

Thinking About Relations

Codd treats relations in a variety of ways in the paper.

- At one level, a relation is simply a *set of tuples*.
- We can also think of each relation as a *predicate* (a function that returns true or false). The relation, when applied to a tuple, returns true only when the tuple belongs to the set.
- We can also think of each relation as a *function* from n - m of the values in the tuple to tuples of the remaining m values.
 - Of course, this is a kind of multiple-valued function. Consider

A	B
0	1
0	2
1	1
3	1

- If we supply 0 as the A parameter, we can get back 1 or 2.

Operations on Relations

- Two basic operations that take relations as parameters and return relations as values
 - *join* - combine two tables in a systematic way to create a new table.
 - *project* - select only certain columns from a table.
- Variations
 - *permute* - projection in which each column appears exactly once.
 - *compose* - combined join and project, typically applicable when we think of relations as functions.
 - ...

Relations as a Programming Language

- If you are convinced that the relational model is a good one (and a *lot* of members of the databases community were so convinced), what should the language you build from it look like?
- Codd had one answer.
- Others had other answers.
- Chamberlin and Boyce provide you with two very different models .
 - In SQUARE, they tried to be very mathematical in form.
 - In SEQUEL, they tried to be much more English-like.
- You need not just the relational operations, but other operations.
- For example, given that many operations returns collections of data (e.g., all salaries), it makes sense

to make such operations a part of the language.

Detour: Describing a Language

- Once you've designed a language, whether it be for relations or for something else. How do you present it to others?
- This paper takes one strategy (we'll discuss it in class).
- Are there other natural strategies?

What's Missing

- Today's thought question: What operations seem to be missing from SEQUEL?