Re: Multiple−Attribute Keys and 1NF

Source: http://newsgroups.derkeiler.com/Archive/Comp/comp.databases.theory/2007−08/msg00426.html

- *From*: "Brian Selzer" <brian@xxxxxxxxxxxxxxxxxxx>
- *Date*: Thu, 30 Aug 2007 13:41:38 −0400

"JOG" <jog@xxxxxxxxxxxxx> wrote in message
news:1188473234.300000.41360@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> On Aug 30, 1:41 am, "Brian Selzer" <br...@xxxxxxxxxxxxxxxxxxx> wrote:
>
>> "JOG" <j...@xxxxxxxxxxxxx> wrote in message
>>
>> news:1188422471.161668.86550@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>
>>> On Aug 29, 7:03 pm, "Brian Selzer"
>>> <br...@xxxxxxxxxxxxxxxxxxx> wrote:
>>>
>>>> "JOG" <j...@xxxxxxxxxxxxx> wrote in
>>>> message
>>>>
>>>> news:1188393382.112445.286350@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>>>
>>>>> On Aug 29, 12:49 pm, Bob
>>>>> Badour
>>>>> <bbad...@xxxxxxxxxxxxxxxx>
>>>>> wrote:
>>>>>
>>>>>> JOG wrote:
>>>>>>
>>>>>>> On
>>>>>>> Aug
>>>>>>> 29,
>>>>>>> 6:10
>>>>>>> am,
>>>>>>> "David
>>>>>>> Cressey"

<cresse...@xxxxxxxxxxx> wrote:

"JOG" <j...@xxxxxxxxxxxxx> wrote in message

news:1188327226.729673.127810@xxxxxxxxxxxxx

Okay, sure. But then to be able to query for green and yellow individually one must employ a further relation encoding two more propositions that state "'Green and yellow' contains 'Green'" and that "'Green and yellow' contains 'Yellow'" respectively. One then has a schema with two domains − one for the composites and one for individual colours (which is what I was inferring when I initially said a new one was being added).

It took me a while to realize that what

you meant from your original description was that "a green and yellow wire means earth". I had thought you meant "a green wire means earth" and "a yellow wire means earth". Pardon me for being dense.

Clearly what we have here is not a domain of colors, but a domain of color codes, where a color code contains one or more colors, and maybe a "thick or thin" qualifier on each color.

It's not clear to me why you need to able to query on simple colors, unless you need to decompose the color coding scheme into its constituent parts for some reason.

There are lot of code domains where each code is made up of a set of more primitive elements. Perhaps a very relevant one might be "character code". If I have the following primitive elements:

B1, B2, B4,

B8, B16, B32, B64, B128 (which might be an odd way of labelling bits 0 through 7 of a byte), I can think of the character code for 'A' as being B64+B1. Now I could query on all the character codes without necessarily having an operator that would yield "all the codes that include B1".

I think that the colors, as constituents of color codes, play the same role as bits, as constituents of character codes. Do you agree?

Yes. I mean no. No, yes. Gnngh ;)

Ok, of course I understand your

point − a wire can be viewed as having a colour code, which itself has constituent parts. But its just one interpretation right. I am still seeing a difference between the propositions:

* There is a colour−code "yellow and green" that denotes "earth".
* The casing of an earth wire features

the colour yellow and the colour green.

Its just like the difference between the propositions:
* My office is B42
* My office is on floor B, room 42.

There are instances where I may well want to encode as the second proposition forms. And

/if/ that were the case (iff), well 1NF is precluding me from doing this in terms of the wire example.

I disagree. You have already noted that 1NF allows this with exactly 2 relations (or with 1 relation and one or more operations on the color code domain.)

True, I do see that, but it does so by requiring the invention of a colour−code concept which isn't in the proposition "The casing of an earth wire features the colour yellow and the colour green".

You have to consider the entire relation
value: what about the
propositions
(treating or exclusively, of course):

"The casing of a live wire features the colour
brown or the colour
red."

"The casing of a neutral wire features the
colour blue or the colour
black."

Write a predicate for the relation schema that
when extentially
quantified
and extended yields a set of atomic formulae
that implies all three of
the
propositions above. I think you'll find that
the colour−code concept
is
in
that predicate.

I agree. I hold little stock with set based values so in RM I
would go
for the addition of colour−code foreign key.

But what if we weren't tied to a traditional relational schema
and
tweaked the system so it could allow propositions with more
than one
role of the same name without decomposing them. As Jan
pointed out
'tuples' are no longer functions − they would be unrestricted
binary
relations (subsets of attribute x values). We could produce a
comparatively simpler and less cluttered schema, predicate in

a very
similar manner as before, and with a few simple alterations
could have
an equally effective WHERE mechanism. My concern
however would be the
consequences to JOIN.

I'm not sure I understand what you are driving at. In the example you
provided, it is the combinations of values from a simple domain that have
significance, regardless of whether they're wrapped in a single attribute
or
not. To me it doesn't make sense to have multiple attributes with the
same
name−−the attribute names correspond to free variables in a predicate:
how
could you assign multiple values to the same variable?

Well consider it this way. If I have the propositions:

The person named Jim speaks the language English
The person named Jim speaks the language German
The person named Brian speaks the language English

I have three propositions, and hopefully we'd agree there are two
roles in these propositions: name and speaks_language. So in FOL I
could write these propositions as:
[P1] Name(x, Jim) −> speaks_language(x, English)
[P2] Name(x, Jim) −> speaks_language(x, English)
[P3] Name(x, Brian) −> speaks_language(x, English)

Are we agreed up to there?

Not exactly. What you have are the roles Name and Language which appear as
free variables in the predicate Speaks. A sentence in FOL is a closed
formula, for example,

exists Name exists Language Speaks(Name,Language)

where each quantifier binds a free variables in Speaks. Supposing that the
domains for Name and Language are,

Names {Jim, Brian, Sue} and
Languages {English, German, French}

respectively, an interpretation of the sentence gives,

Speaks(Jim,English) /\
Speaks(Jim,German) /\

~Speaks(Jim,French) /\
Speaks(Brian,English) /\
~Speaks(Brian,German) /\
~Speaks(Brian,French) /\
~Speaks(Sue,English) /\
~Speaks(Sue,German) /\
~Speaks(Sue,French)

Which under the closed world assumption becomes,

Speaks(Jim,English) /\
Speaks(Jim,German) /\
Speaks(Brian,English)

From this it can be deduced that Jim speaks both English and German, and

that Jim and Brian both speak English. Under the domain closure assumption, it can be deduced that Sue does not exist, and that the only languages that exist are English and German. Sue can exist and French can exist, but since neither are referenced, neither does. It should be noted that just because Brian exists and German exists doesn't mean that Brian speaks German. The truth value for Speaks(Brian,German) was assigned false under the given interpretation.

> If so then [P1] ^ [P2] gives us (via
> composition):
> Name(x, Jim) −> speaks_language(x, English) ^ speaks_language(x,
> English)
>
> and we are left with a sentence that has two distinct roles, one of which appears twice. All of this sort of thinking has been driven by a distaste us having to add a magic 'header' component to a relation (probably as a consequence of reading pascal's work), and the desire to bring roles back into the equation.
>
>> But you can
>> certainly assign a set of values to a variable that expects a set of
>> values,
>> since a set is a value! And you can certainly have a predicate with free variables that range over relations and free variables that range over individuals−−it's just that the predicate is no longer first order.

.