Database Design Lookup tables

Asked 16 years, 3 months ago Modified 15 years, 3 months ago

Viewed 2k times

4

I'm currently trying to improve the design of a legacy db and I have the following situation

Currently I have a table SalesLead in which we store the the LeadSource.

```
Create Table SalesLead(
    ....
    LeadSource varchar(20)
    ....
)
```

The Lead Sources are helpfully stored in a table.

```
Create Table LeadSource (
    LeadSourceId int,   /*the PK*/
    LeadSource varchar(20)
)
```

And so I just want to Create a foreign key from one to the other and drop the non-normalized column.

All standard stuff, I hope.

Here is my problem. I can't seem to get away from the issue that instead of writing

```
SELECT * FROM SalesLead Where LeadSource = 'foo'
```

Which is totally unambiguous I now have to write

```
SELECT * FROM SalesLead where FK_LeadSourceID = 1
```

or

```
SELECT * FROM SalesLead
INNER JOIN LeadSource ON SalesLead.FK_LeadSourceID
= LeadSource.LeadSourceId
where LeadSource.LeadSource = "foo"
```

Which breaks if we ever alter the content of the LeadSource field.

In my application when ever I want to alter the value of SalesLead's LeadSource I don't want to update from 1 to 2 (for example) as I don't want to have developers having to remember these **magic numbers**. The ids are arbitrary and should be kept so.

How do I remove or negate the dependency on them in my app's code?

Edit Languages my solution will have to support

- .NET 2.0 + 3 (for what its worth asp.net, vb.net and c#)
- vba (access)
- db (MSSQL 2000)

Edit 2.0 The join is fine is just that 'foo' may change on request to 'foobar' and I don't want to haul through the queries.

database-design magic-numbers

Share

Improve this question

Follow

edited Sep 18, 2009 at 16:16

GameFreak

2,921 ● 7 ● 34 ● 38

asked Sep 17, 2008 at 9:42

NaughtyNaughty

7 Answers

Sorted by: Highest score (default)

3

If you want to de-normalize the table, simply add the LeadSource (Varchar) column to your SalesLead table, instead of using a FK or an ID.

On the other hand, if your language has support for ENUM structures, the "magic numbers" should be safely stored in an enum, so you could:

```
SELECT * FROM SALESLEAD WHERE LeadSouce = (int)
EnmLeadSource.Foo; //pseudocode
```

And your code will have a

```
public enum EnmLeadSource
{
   Foo = 1,
   Bar = 2
}
```

It is OK to remove some excessive normalization if this causes you more trouble than what it fixes. However, bear in mind that if you use a VARCHAR field (as oposed to a Magic Number) you must maintain consistency and it could be hard to localize later if you need multiple languages or cultures.

The best approach after Normalization seems to be the usage of an Enum structure. It keeps the code clean and you can always pass enums across methods and functions. (I'm assuming .NET here but in other languages as well)

Update: Since you're using .NET, the DB Backend is "irrelevant" if you're constructing a query through code. Imagine this function:

```
public void GiveMeSalesLeadGiven( EnmLeadSource
thisLeadSource )
{
  // Construct your string using the value of
thisLeadSource
}
```

In the table you'll have a LeadSource (INT) column. But the fact that it has 1,2 or N won't matter to you. If you later need to change foo to foobar, that can mean that:

1) All the "number 1" have to be number "2". You'll have to update the table. 2) Or You need Foo to now be number 2 and Bar number 1. You just change the Enum (but make sure that the table values remain consistent).

The Enum is a very useful structure if properly used.

Hope this helps.

Share Improve this answer Follow

edited Sep 17, 2008 at 10:20

answered Sep 17, 2008 at 9:50

Martin Marconcini

27.2k ● 19 ● 108 ● 148

2

Have you considered just not using an artificial key for the `LeadSource` table? Then you get to use `LeadSource` as the FK in `SalesLead`, which simplifies your queries while retaining the benefits of using a canonical set of values (the rows in `LeadSource`).

Share Improve this answer Follow

answered Sep 17, 2008 at 12:34
Hank Gay
71.8k ● 36 ● 161 ● 222

Hank's answer is a good one. I would add that the use of LeadSource is *not* denormalizing, because it looks like that is the real code. Add a descriptive column to the lookup table and change that when you need to. – Ken Downs Jan 17, 2011 at 4:33

1

Did you consider an updatable view? Depending on your database server and the integrity of your database design you will be able to create a view that, when its values change, in turn it will update the constituent tables.

Share Improve this answer Follow

answered Sep 17, 2008 at 9:47
LohanJ
45 ● 1

0

I really don't see your problem behind the join.

Naturally, asking directly by the FK_LeadSourceID is wrong, but using the JOIN seems to be the right way to go as I masks changing IDs perfectly fine. If, for example, "foo" becomes 3 at one day (and you update the foreign key field), the last query you've displayed will still work exactly the same.

If you want to make the change to the schema without altering the current queries in the application, then a view encompassing this join is the way to go.

Or if you fear that the join Syntax is non-intuitive, there's always the subselect...

```
SELECT * FROM SalesLead where FK_LeadSourceID =
         (SELECT LeadSourceID from LeadSource
WHERE LeadSource = 'foo')
```

but remember to keep an index on LeadSource.LeadSource - at least if you have a lot of them stored in the table.

Share Improve this answer

Follow

answered Sep 17, 2008 at 9:51

pilif

12.7k ● 5 ● 36 ● 31

the issue is that when if foo is changed to foobar then all my queries will need updating – NaughtyNaughty Sep 17, 2008 at 9:56

If you "improve design" by introducing new relations/tables, you'll certainly have the need for different entities. If so, you'll need to deal with their semantics.

0

In the previous solution you were able to just update the LeadSource name to whatever you wanted in the appropriate SalesLead row. If you update the name in your new structure, you do so for all SalesLead rows.

There is no way around dealing with these different semantics. You just have to do so. In order to make the tables easier to query, you might use views as already suggested, but I'd expect them mostly for reporting purposes or backward compatibility, provided they are not updatable, because everybody updating this view would not be aware of changed semantics.

If you dislike the join try SELECT * FROM SalesLead where LeadSourceId IN (SELECT Id FROM LeadSource WHERE LeadSource = 'foo')

Share Improve this answer Follow

answered Sep 17, 2008 at 9:58

Olaf Kock
48k ● 9 ● 62 ● 91

This is the thought behind the improved design. If get all my leads from 'Joel' who changes his name to 'Jeff' then I would want all leads that were from Joel to be updated to 'Jeff' – NaughtyNaughty Sep 17, 2008 at 10:11

0

In a typical application the user would be presented with a list of Lead Sources (returned by querying the LeadSource table) and the subsequent SalesLead query would be dynamically created by the application based upon the user's selection.

Your application appears to have some 'well known' lead sources that you need to write specific queries for. If this is the case, then add a third (unique) field to the LeadSource table that includes an invariant 'name' that you can use as the basis of your application's queries.

This shifts the burden of magic-ness from a DB generated magic number (that may vary from installation to installation) to a system defined magic name (that is fixed by design).

Share Improve this answer Follow

answered Sep 17, 2008 at 10:01

Seb Rose
3,666 ● 20 ● 29

0

There's a false dichotomy here.

```
SELECT * FROM SalesLead
INNER JOIN LeadSource ON SalesLead.FK_LeadSourceID
= LeadSource.LeadSourceId
where LeadSource.LeadSource = "foo"
```

doesn't break any more than the original

```
SELECT * FROM SalesLead Where LeadSource = 'foo'
```

when `foo` changes to `foobar`. Also, if you're using parameterized queries (and you really should be), you don't have to change anything when `foo` changes to `foobar`.

Share Improve this answer Follow

answered Sep 17, 2008 at 11:29

Hank Gay
71.8k ● 36 ● 161 ● 222

"doesn't break any more than the original" - agreed
– NaughtyNaughty Sep 17, 2008 at 12:22