What's the best database structure to keep multilingual data? [duplicate]

Asked 10 years ago Active 11 months ago Viewed 43k times

54

32

This question already has answers here:

Closed 8 years ago.

Possible Duplicate:
Schema for a multilanguage database

Here's an example:

```
[ products ]
id (INT)
name-en_us (VARCHAR)
name-es_es (VARCHAR)
name-pt_br (VARCHAR)
description-en_us (VARCHAR)
description-es_es (VARCHAR)
description-pt_br (VARCHAR)
price (DECIMAL)
```

The problem: every new language will need modify the table structure.

Here's another example:

```
[ products-en_us ]
id (INT)
name (VARCHAR)
description (VARCHAR)
price (DECIMAL)

[ products-es_es ]
id (INT)
name (VARCHAR)
description (VARCHAR)
price (DECIMAL)
```

The problem: every new language will need the creation of new tables and the "price" field is duplicated in every table.

Here's another example:

```
[ languages ]
id (INT)
name (VARCHAR)

[ products ]
id (INT)
price (DECIMAL)

[ translation ]
id (INT, PK)
model (VARCHAR) // product
field (VARCHAR) // name
language_id (INT, FK)
text (VARCHAR)
```

The problem: hard?

mysql database localization translation multilingual

edited Apr 3 '19 at 16:13

TylerH
17.5k 12 58 74

asked Feb 9 '10 at 9:31

Thiago Belem
7,178 2 38 63

5 The third method is more or less correct - what's hard about it? – K Prime Feb 9 '10 at 9:45

2 The problem is, that with every solution you find, you'll always find a case, when you need to modify table - i.e. more languages, different languages, another field... – Adam Kiss Feb 9 '10 at 9:57

You can check this link: gsdesign.ro/blog/multilanguage-database-design-approach although reading the comments is very helpful – Fareed Alnamrouti Jan 18 '12 at 13:54

8 Answers

Your third example is actually the way the problem is usually solved. Hard, but doable.

28

Remove the reference to product from the translation table and put a reference to translation where you need it (the other way around).

```
[ products ]
id (INT)
price (DECIMAL)
title_translation_id (INT, FK)

[ translation ]
id (INT, PK)
neutral_text (VARCHAR)
-- other properties that may be useful (date, creator etc.)

[ translation_text ]
translation_id (INT, FK)
language_id (INT, FK)
text (VARCHAR)
```

As an alternative (not especially a good one) you can have one single field and keep all translations there merged together (as XML, for example).

```
<translation>
  <en>Supplier</en>
  <de>Lieferant</de>
  <fr>Fournisseur</fr>
</translation>
```

edited Feb 9 '10 at 9:40

answered Feb 9 '10 at 9:32
user151323

11 What if the product table contains several translated fields ? When retrieving products, you will have to do one additional join per translated field, which will result in severe performance issues. There is as well (IMO) additional complexity for insert/update/delete. The single advantage of this is the lower number of tables. I would go for the method proposed by Gipsy King or Clément : I think it's a good balance between performance, complexity, and maintenance issues. – Frosty Z Jan 20 '11 at 10:05

39

Similar to method 3:

```
[languages]
id (int PK)
```

```
code (varchar)

[products]
id (int PK)
neutral_fields (mixed)

[products_t]
id (int FK)
language (int FK)
translated_fields (mixed)
PRIMARY KEY: id,language
```

So for each table, make another table (in my case with "_t" suffix) which holds the translated fields. When you `SELECT * FROM products`, simply `... LEFT JOIN products_t ON products_t.id = products.id AND products_t.language = CURRENT_LANGUAGE`.

Not that hard, and keeps you free from headaches.

edited Dec 14 '14 at 12:00

DanMan
9,555 3 32 56

answered Feb 9 '10 at 15:28

Gipsy King
1,343 8 15

14

In order to reduce the number of JOIN's, you could keep separate the translated and non translated in 2 separate tables :

```
[ products ]
id (INT)
price (DECIMAL)

[ products_i18n ]
id (INT)
name (VARCHAR)
description (VARCHAR)
lang_code (CHAR(5))
```

answered Feb 9 '10 at 9:51

Clément
6,110 1 26 22

@Clément - The problem here's is when products table get a new field... I'll need to change the products_i18n table too. :/ – Thiago Belem Feb 9 '10 at 9:53

7 @TiuTalk - only one of the table will get the new field, if it's a translated field, it goes into `products_i18n` , otherwise it goes in `products` . This way you don't duplicate any information. – Clément Feb 9 '10 at 10:04

@Clément: product.id is user as FK in products_i18n.id or you use third join table? – CoR Dec 12 '15 at 18:59

@CoR Yes, `products.id` could be a foregin key in the `products_i18n` table. The primary key of the `products_i18n` table would be a composite key composed of both `(product.id, products_i18n.lang_code)` . – War10ck Oct 10 '16 at 15:52

3

At my $DAYJOB we use gettext for I18N. I wrote a plugin to xgettext.pl that extracts all English text from the database tables and add them to the master messages.pot.

It works very well - translators deal with only one file when doing translation - the po file. There's no fiddling with database entries when doing translations.

answered Feb 9 '10 at 15:12

holygeek
13.3k 1 33 46

This may work if you only want to provide translations for your application. F.ex. Menu Entries, Headlines, Helptexts etc. – widdy May 7 '19 at 14:16

2

[languages] id (int PK) code (varchar)

```
[products]
id (int PK)
name
price
all other fields of product
id_language ( int FK )
```

I actually use this method, but in my case, it's not in a product point of view, for the various pages in my CMS, this work's quite well.

If you have a lot of products it might be a headache to update a single one in 5 or 6 languages... but it's a question of working the layout.

answered Feb 9 '10 at 17:33

Tio
456 1 8 17

What about fourth solution?

0

```
[ products ]
id (INT)
language (VARCHAR 2)
name (VARCHAR)
description (VARCHAR)
price (DECIMAL)
*translation_of (INT FK)*
```

Translation_of is **FK** of it self. When You add default language *translation_of* is set to Null. But when you add second language *translation_of* takes primary produkt language id.

```
SELECT * FROM products WHERE id = 1 AND translation_of = 1
```

In that case we get all translations for product with id is 1.

```
SELECT * FROM products WHERE id = 1 AND translation_of = 1 AND language = 'pl'
```

We get only product in Polish translation. Without second table and JOINS.

edited Sep 4 '11 at 14:51

answered Sep 4 '11 at 14:16

chf

107 1 3

That's an interesting approach. I like the ease of querying but it does break the assumption that ever entry in the product table is one product, so one has to keep that in mind. It also allows you to keep using the proper types (varchar, etc.) for the fields. – Rolf Feb 28 '15 at 22:31

1 I was thinking of implementing the exact same thing right now, but haven't found this solution anywhere else. I see your post is from 2011. Have you had any problems with this? Do you still think it's a good solution? Thanks. – tabacitu Sep 7 '15 at 16:33

1 1) It allows you to have both translatable and non-translatable fields; – tabacitu Dec 12 '15 at 19:20

1 2) It does not tie all of your translations to a specific one (the default language); – tabacitu Dec 12 '15 at 19:21

2 An to answer your question, the inserts, updates, deletes are doing fine and snappy. The problem is all in my head - I know the DB isn't as it should be and that bugs me. – tabacitu Dec 12 '15 at 19:23

-1

Have many to many relationship.

You have your data table, languages table and a data_language table.

In the data_language table you have

> id, data_id, language_id

I think that might work best for your.

answered Feb 9 '10 at 9:36

AntonioCS
7,462 17 57 85

@AntonioCS - The "data" table isn't the "product" table, right? – Thiago Belem Feb 9 '10 at 9:37

@TiuTalk it is. This way the product table doesn't have to know which languages there are, neither does the language table. It's all on the data_language table (or in this case 'product_language table) – AntonioCS Feb 9 '10 at 10:30

-2

We use this concept for our webiste (600k views per day) and (maybe surprisingly) it works. Sure along with caching and query optimalization.

```
[attribute_names]
id (INT)
name (VARCHAR)

[languages_names]
id (INT)
name (VARCHAR)

[products]
id (INT)
attr_id (INT)
value (MEDIUMTEXT)
lang_id (INT)
```

answered Feb 9 '10 at 9:54

Petr Peller
7,556 8 44 62

And what about duplicated fields like price? – Thiago Belem Feb 9 '10 at 10:42