What are best practices for multi-language database design? [closed]

Asked 12 years, 3 months ago Active 6 years, 9 months ago Viewed 145k times

211

122

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 9 years ago.

What is the best way to create multi-language database? To create localized table for every table is making design and querying complex, in other case to add column for each language is simple but not dynamic, please help me to understand what is the best choose for enterprise applications

sql database database-design

Share Edit Follow

edited Oct 17 '11 at 19:57
Dan J
15.6k 6 47 78

asked May 30 '09 at 9:17
Arsen Mkrtchyan
47.6k 30 144 178

See as well : stackoverflow.com/questions/2227985/... stackoverflow.com/questions/316780/... stackoverflow.com/questions/3077305/... – Frosty Z Jan 20 '11 at 10:20

4 I've found this very helpful to me: Multilanguage Database Design in MySQL – Siamak A.Motlagh Jul 7 '14 at 7:27

5 Answers

Active Oldest Votes

What we do, is to create two tables for each multilingual object.

250

E.g. the first table contains only language-neutral data (primary key, etc.) and the second table contains one record per language, containing the localized data plus the ISO code of the language.

In some cases we add a DefaultLanguage field, so that we can fall-back to that language if no localized data is available for a specified language.

Example:

```
Table "Product":
----------------
ID                 : int
<any other language-neutral fields>


Table "ProductTranslations"
---------------------------
ID                 : int      (foreign key referencing the Product)
Language           : varchar  (e.g. "en-US", "de-CH")
IsDefault          : bit
ProductDescription : nvarchar
<any other localized data>
```

With this approach, you can handle as many languages as needed (without having to add additional fields for each new language).

Update (2014-12-14): please have a look at this answer, for some additional information about the implementation used to load multilingual data into an application.

Share Edit Follow

edited May 23 '17 at 11:33
Community ♦
1 1

answered May 30 '09 at 9:28
M4N
91k 45 211 256

15 what if the only language-neutral field is the id? and how exactly do you insert the foreign key reference when inserting a row? – Timo Huovinen Sep 18 '10 at 18:16

4 It's funny, that I was designing a database scheme for a multilingual CMS and had this question in my head aswell. I chose this approach, before I even saw this answer! Thanks for this answer! – Patrick Manser May 29 '13 at 8:50

6 One thing to note here is that there would either be no PK on this table or ID and Language need to be the composite PK. Either that, or you need to

add a ProductTranslationId field, probably as an identity. – Daniel Lorenz Aug 6 '14 at 19:38

1 @Luca: I answered your question, showing what implementation(s) I use to load the data. – M4N Dec 15 '14 at 10:41

1 @AarónGutiérrez Well, funnily enough you create a table with a single column called `id` :D . To explain, each `id` represents a meaning to which you can attach words from any language in a relational table, so you get two tables, `meaning` (id) and `word` (id, meaning_id), the `id` in the `word` table represents the word id, the `id` in the `meaning` represents the meaning that is universal. – Timo Huovinen Sep 17 '18 at 19:27

21

I recommend the answer posted by Martin.

But you seem to be concerned about your queries getting too complex:

> To create localized table for every table is making design and querying complex...

So you might be thinking, that instead of writing simple queries like this:

```
SELECT price, name, description FROM Products WHERE price < 100
```

...you would need to start writing queries like that:

```
SELECT
  p.price, pt.name, pt.description
FROM
  Products p JOIN ProductTranslations pt
  ON (p.id = pt.id AND pt.lang = "en")
WHERE
  price < 100
```

Not a very pretty perspective.

But instead of doing it manually you should develop your own database access class, that pre-parses the SQL that contains your special localization markup and converts it to the actual SQL you will need to send to the database.

Using that system might look something like this:

```
db.setLocale("en");
db.query("SELECT p.price, _(p.name), _(p.description)
          FROM _(Products p) WHERE price < 100");
```

And I'm sure you can do even better that that.

The key is to have your tables and fields named in uniform way.

Share Edit Follow

answered May 30 '09 at 10:37
Rene Saarsoo
12.8k 8 53 77

the other question is, to create one business object for product? or to create two... in first case it's easy to work with that item, in 2 nd easy to write CMS – Arsen Mkrtchyan May 30 '09 at 10:52

15

I find this type of approach works for me:

```
Product      ProductDetail         Country
=========    ==================    =========
ProductId    ProductDetailId       CountryId
- etc -      ProductId             CountryName
             CountryId             Language
             ProductName           - etc -
             ProductDescription
             - etc -
```

The ProductDetail table holds all the translations (for product name, description etc..) in the languages you want to support. Depending on your app's requirements, you may wish to break the Country table down to use regional languages too.

Share Edit Follow

edited May 30 '09 at 9:40

answered May 30 '09 at 9:33
Nick
5,508 9 51 72

I chose this same approach for a project I'm currently working on because my different locales contain very specific information about unit systems and measures to be displayed to users. – califrench Apr 3 '14 at 2:24

10 Country and language (locales) are different things. And ISO language codes are natural keys, you eliminate unnecessary join from lang to country. – gavenkoa Jun 4 '14 at 6:01

11

I'm using next approach:

Product

ProductID OrderID,...

ProductInfo

ProductID Title Name LanguageID

Language

LanguageID Name Culture,....

Share Edit Follow

answered May 30 '09 at 9:38
omoto
1,202 1 17 24

2

Martin's solution is very similar to mine, however how would you handle a default descriptions when the desired translation isn't found ?

Would that require an IFNULL() and another SELECT statement for each field ?

The default translation would be stored in the same table, where a flag like "isDefault" indicates wether that description is the default description in case none has been found for the current language.

Share Edit Follow

answered Aug 24 '09 at 14:39

doM
51 2

1 @GorrillaApe: see this answer for an example how to fall back to the default language, if the desired language was not found: stackoverflow.com/a/27474681/19635 – M4N Apr 21 '17 at 8:38