Language Text holds the “text” of any field from any table, and it holds the text in one or more languages.

**Design 1**

Take for example the Environment table which has just one multi-language field, namely Description. We use the text field of the LanguageText table to hold the various translated values. For example the Description field is set to “home”, and we want “home” to be in three languages namely English, Spanish, and German. We first have to add the three languages to the Name field of the Language table. Next we have to add “Environment” to the Tablename field and “Description” to Field1 of the LanguageRecipientTable table. The LanguageRecipientTable table connects the Enviroment table (or any other indicated table) to the LanguageText table. Finally we take the Enviroment table ID from the LanguageRecipientTable and store it in the LangugageRecipientTableID field of the LanguageText table, along with the row id of the “home” record from the Enviroment table (this goes into RecipientRowID field) and we set LanguageRecipientFieldNumber to 1 to indicate this record is for the “Description” field of the “Enviroment” table (i.e Field1 in LanguageRecipientTable is set to Description). We repeat this partial row three times. To the first partial row we add the LanguageID from the Language Table that corresponds to English, and “Home” to the Text field. To the second partial row we add the LanguageID from the Language Table that corresponds to Spanish, and “Casa” to the Text field. To the third partial row we add the LanguageID from the Language Table that corresponds to German, and “Hause” to the Text field.

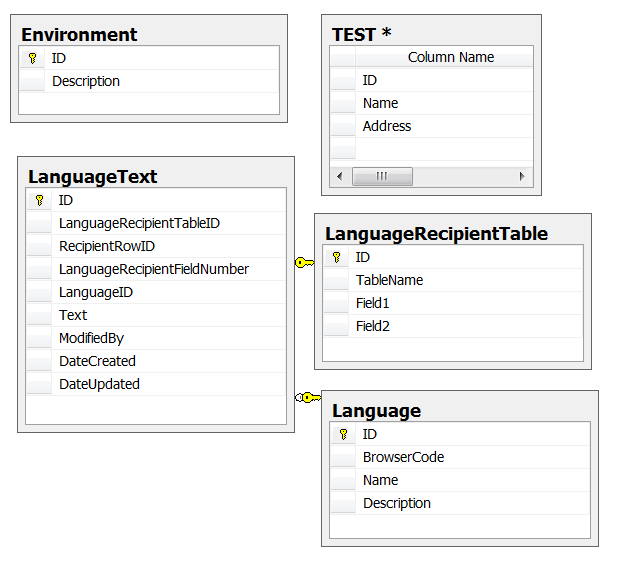
In comparison the TEST table has 2 fields that need to be held in the three languages. So we do the same thing but in the LanguageRecipientTable table we put “Name” in Field1 and “Address” in Field2. Then we add six records to the LanguageText table with the first three having a 1 for LanguageRecipientFieldNumber and the second three have 2 for LanguageRecipientFieldNumber (1 = Name, 2 = Address from LanguageRecipientTable).

If we had a table that need three fields held in multiple languages we’d just add a Field3 to the LanguageRecipientTable table.

Essential LangugageRecipientTableID, RecipientRowID, LanguageRecipientFieldNumber, and LanguageID are a compound primary key of LanguageText. ModifiedBy, DateCreated, DateUpdated are option fields of LanguageText. BrowserCode, and Description are optional fields of the Language table.

LanguageText essentially is an extension of any table (in this example, an extension of the Environment table), and it is tied to another table via the ID from the other table (ie Environment table) and the RecipientRowID field in LanguageText. Since LanguageText is an extension of multiple tables and since any of those table might have the save row IDs as some or all of the other tables we use the compound primary key mentioned above to make the connection unique.

It’s also worth noting that the recipient tables really shouldn’t have fields in their design for the fields LanguageText represents for them. It’s easier to design a table with the fields it needs, then decided which need to be in multiple languages and then move those fields to LanguageText and name those fields in LanguageRecipientTable.



This design (design 1), allows every multiple language field to be tracked in regards to ModifiedBy, DateCreated, and DateUpdated. Another reason to use this design is when the translated text is large (ie varchar(max)). The draw back to this LangugageRecipientTableID, RecipientRowID, LanguageRecipientFieldNumber, and LanguageID have to be repeated for each field, even though they are shared for a recipient table that has multiple fields in LanguageText. Another draw back of this design is if one wants to get back the text for two or more fields (like we might need to do for the Test table). You would have to do two separate queries with two different LanguageRecipientFieldNumber. Design 2, example below has a corresponding row for each TestTable, allowing a single query to get back the text for each field. In so doing it gives up the tracking on each field, and instead tracks changes by row.

To Select “text” value from LanguageText for a particular table and field use this proc:

CREATE PROCEDURE [dbo].[LanguageText\_Select\_ByLanguageID\_RecipientTableID\_RecipientFieldNumber]

@LanguageID int

,@RecipientTableID int

,@RecipientFieldNumber int

AS

BEGIN

DECLARE @sql AS varchar(max)

SET @sql = 'Insert into #tempFieldName Select Field{FIELDNUMBER} from dbo.LanguageRecipientTable where (ID = {TableID})'

SET @sql = REPLACE(@sql, '{FIELDNUMBER}', @RecipientFieldNumber)

SET @sql = REPLACE(@sql, '{TableID}', @RecipientTableID)

Create Table #tempFieldName(fieldName nVarChar(50))

EXEC (@sql)

Declare @RecipientField nvarchar(50)

Set @RecipientField = (Select \* from #tempFieldName)

SELECT

dbo.LanguageText.RecipientRowID

,dbo.LanguageText.[Text]

,dbo.Language.Name AS Language

,dbo.LanguageRecipientTable.TableName as 'RecipientTable'

,@RecipientField as 'RecipientField'

FROM dbo.LanguageText INNER JOIN

dbo.Language ON dbo.LanguageText.LanguageID = dbo.Language.ID INNER JOIN

dbo.LanguageRecipientTable ON dbo.LanguageText.LanguageRecipientTableID = dbo.LanguageRecipientTable.ID INNER JOIN

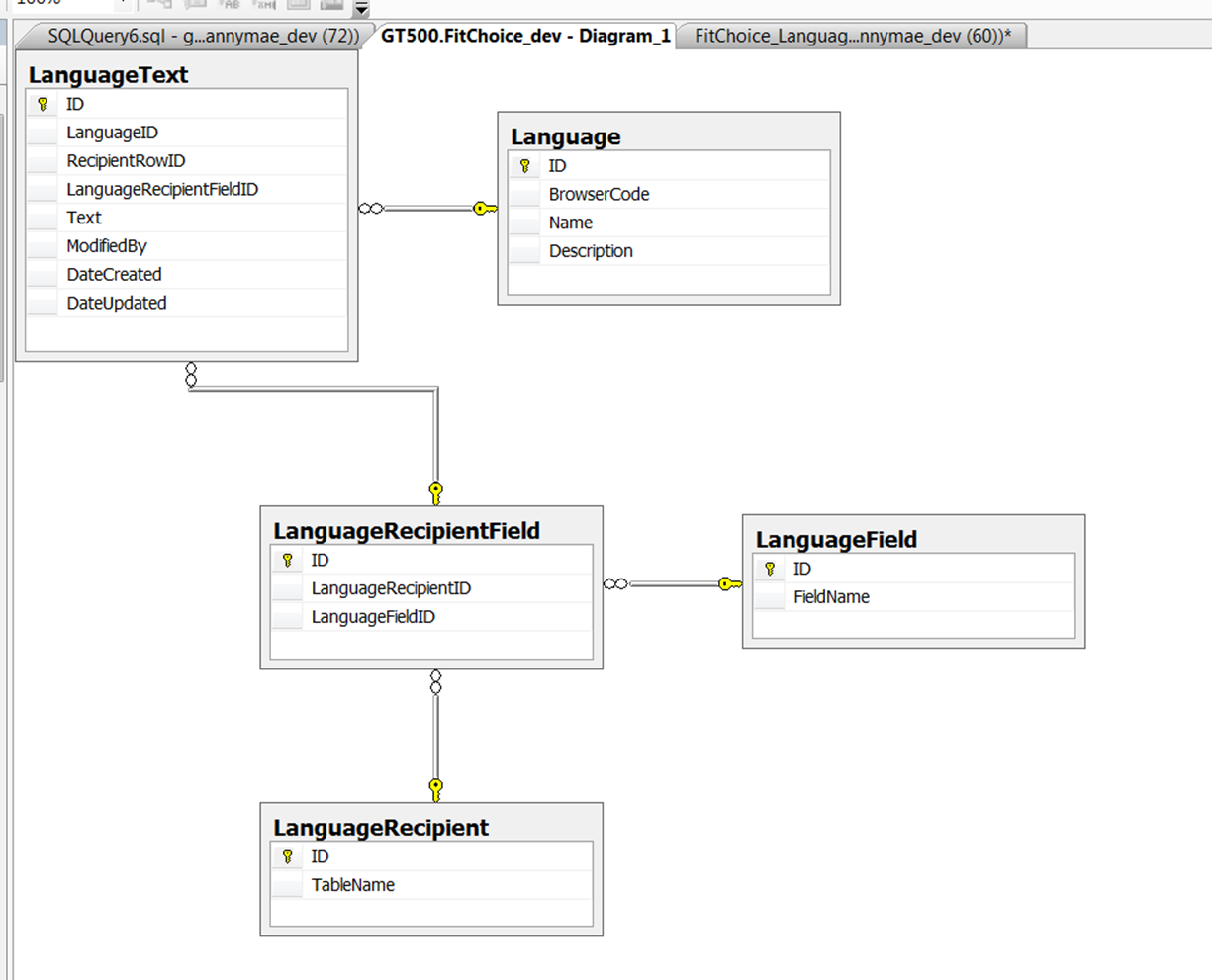
dbo.Environment ON dbo.LanguageText.RecipientRowID = dbo.Environment.ID

WHERE (dbo.LanguageText.LanguageID = @LanguageID)

AND (dbo.LanguageText.LanguageRecipientTableID = @RecipientTableID)

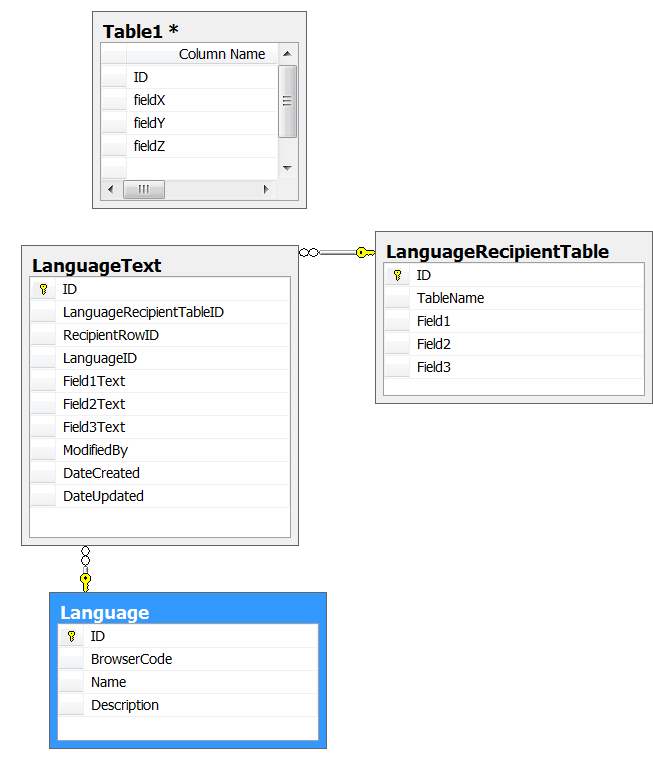
and (dbo.LanguageText.LanguageRecipientFieldNumber = @RecipientFieldNumber)

Design 1a – with unlimited Fields. Here we a field table (LanguageField) to hold field names instead of having them as part of the LanguageRecipientTable , and we add a LanguageRecipientField table that ties a fieldname to a Recipient Table Name, and we replace LanguageRecipientFieldNumber with the ID from the LanguageREcipientField table. This design works the same as Design1, but makes adding a new field easier as no tables need to be redesigned. It’s not easy to select items as a row with this design, but see design 3, which will give you an ideas on how that could possibly be accomplished.



**Design 2.**

With design 2 all multi-language fields from the recipient table (ie Table1, in diagram below) are saved in a single row in the LanguageText table, as such there is no need for LanguageRecipientFieldNumber, and the Text field has been replaced with Field1Text, Field2Text, Field3Text. If more than three fields per row will need multiple language simply add more FieldXText columns to LanguageText and more FieldX columns to LanguageRecipientTable. Tracking, such as ModifiedBy, is now done on a per row basis as opposed to a per field basis in Design 1. Again, there is no need to have fieldX,Y,and Z in Table1, as LanguageText really holds there value and LanguageRecipientTable defines there names.



To Select “text” values from LanguageText for a particular recipient table use this sql:

SELECT

dbo.LanguageText.RecipientRowID

,dbo.LanguageRecipientTable.TableName

, dbo.LanguageRecipientTable.Field1Name

, dbo.LanguageText.Field1Text

, dbo.LanguageRecipientTable.Field2Name

, dbo.LanguageText.Field2Text

, dbo.LanguageRecipientTable.Field3Name

, dbo.LanguageText.Field3Text

, dbo.Language.Name as [Language]

FROM dbo.LanguageText

INNER JOIN dbo.Language

ON dbo.LanguageText.LanguageID = dbo.Language.ID

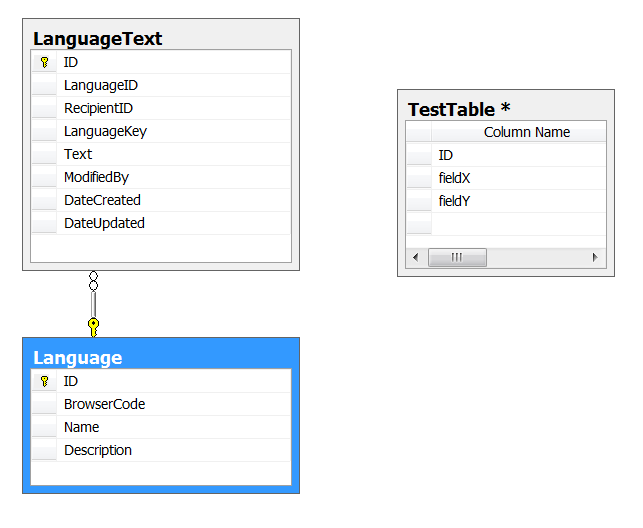
INNER JOIN dbo.LanguageRecipientTable

ON dbo.LanguageText.LanguageRecipientTableID = dbo.LanguageRecipientTable.ID

WHERE (dbo.LanguageText.LanguageID = 78)

AND (dbo.LanguageText.LanguageRecipientTableID = 2)

To limit the above sql to just a row add: AND dbo.LanguageText.RecipientRowID = 1, to the WHERE clause.

Design 3 - This is essentially the same as design 1 but it gets the design down to just two tables. The LanguageRecipientField, LanguageField and LanguageRecipient Tables of design 1 a are no longer needed, instead it relies on the data enterer to stick to a standard of TableName.FieldName for the LanguageKey field of LanguageText table. LanguageKey field is a replacement for LanguageRecipientFieldID in the LanguageText table of design 1a. For example if TestTable has two fields (namely fieldX and fieldY) who’s text is stored in LanguageText the LanguageKey would be “TestTable.fieldX” and “TestTable.fieldY”. From those Keys we have the name of the Recipient Table and the Recipient Field. This design is a lot more human readable than design 1a, but it is the same in all practicality. Design 1, has more referential integrity, and thus eliminates the potential for human errors such as miss typed LanguageKey values (which is really the only down side to design 3).

For design 3, use this stored procedure to get back all of the fields and rows for one table in a particular language:

USE [FitChoice\_dev]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[LanguageText\_SelectByTableNameAndLanguage] Script Date: 04/26/2012 09:02:55 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[LanguageText\_SelectByTableNameAndLanguage]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This proc selects all rows and fields for a particular table

in a particular language

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@RecipientTableName AS nvarchar(250) , -- ie Environment, or Goal, or whatever table is storing it's values in the LanguageText table

--It is assumed LanguageText.LanguageKey is in the format TableName.Field, as in Enviroment.Description

@LanguageID as int

AS

BEGIN

if CHARINDEX('.', @RecipientTableName) < 1

Begin

--Add the separator to the TableName

Set @RecipientTableName = @RecipientTableName + '.'

end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--This section is useful when debugging the Stored Proc

drop table #temp6

Declare @RecipientTableName AS nvarchar(250)

Declare @LanguageID as int

Set @RecipientTableName = 'Environment.'

Set @LanguageID =50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--Step 1. Create a temp table to hold the field names for the passed in RecipientTableName and Language

CREATE TABLE #temp6 (

ID INT IDENTITY(1, 1)

,field nvarchar(250) )

--Step 2. Generate the SQL that will get the field names and save them to the temp table

DECLARE @sqlT AS varchar(max)

set @sqlT = 'Insert into #temp6(field) select LanguageKey from LanguageText where [Languagekey] like '''

+ @RecipientTableName

+ '%'' and LanguageID = ' + Cast(@LanguageID as varchar(200)) + ''

Exec(@sqlT)

--Step 3. Get a count of the number of fields

DECLARE @FieldCount INT

select @FieldCount = count(ID)from #temp6

--Step 4. Declare and initialize variables

DECLARE @sql AS varchar(max)

SET @sql = ''

DECLARE @FieldName AS varchar(max)

SET @FieldName = ''

Declare @LagFieldName as nvarchar(250)

set @LagFieldName = ''

--print @sql

--Step 5. Loop through the temp table making the list of fields to be selected

WHILE @FieldCount > 0

BEGIN

set @FieldName = ''

select @FieldName = field from #temp6 where id = @FieldCount

if(@FieldName != @LagFieldName) --if we already have this field don't get it again

begin

SET @sql = @sql + ' ,Max( Case When LanguageKey = ''' + @FieldName + ''' then [Text] end) as [' + replace(@FieldName, @RecipientTableName, '') + ']'

end

set @FieldCount = @FieldCount - 1

set @LagFieldName = @FieldName --save the last field name

End

--Step 6. Drop the temp table

drop table #temp6

--Step 7. Finish the SQL

DECLARE @FinishSQL AS varchar(max)

Set @FinishSQL = ' from (Select RecipientID, LanguageKey, [TEXT] from LanguageText '

Set @FinishSQL = @FinishSQL + ' where LanguageID = ' + Cast(@LanguageID as varchar(200)) + ' and [Languagekey] in( '

Set @FinishSQL = @FinishSQL + ' Select LanguageKey from LanguageText '

Set @FinishSQL = @FinishSQL + ' where [Languagekey] like ''' + @RecipientTableName + '%'' and LanguageID = ' + Cast(@LanguageID as varchar(200)) + ')'

Set @FinishSQL = @FinishSQL + ') as TTT group by RecipientID'

--Step 8. Make and run the complete SQL

set @sql = 'select RecipientID' + @sql + @FinishSQL

exec(@sql)

END

/\*

The KEY to this proc is:

select RecipientID as EnviromentRowID

,Max( Case When LanguageKey = 'Environment.Description' then [Text] end) as [Description]

,Max( Case When LanguageKey = 'Environment.Test' then [Text] end) as [Test]

from

(

Select RecipientID, LanguageKey, [TEXT] from LanguageText where [Languagekey]

in( Select LanguageKey from LanguageText where [Languagekey] like 'Environment.%' and RecipientID = 1)

) as a

group by RecipientID

\*/

GO

For design 3, use this stored procedure to get back all of the fields for a particular row from one table in a particular language:

USE [FitChoice\_dev]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[LanguageText\_SelectByTableNameLanguageAndRow] Script Date: 04/26/2012 09:02:17 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[LanguageText\_SelectByTableNameLanguageAndRow]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This proc selects all fields for a particular row of a particular table

in a particular language

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@RecipientTableName AS nvarchar(250) , -- ie Environment, or Goal, or whatever table is storing it's values in the LanguageText table

--It is assumed LanguageText.LanguageKey is in the format TableName.Field, as in Enviroment.Description

@RecipientRowID as INT, --The Row ID from the Recipient Table ie Environment

@LanguageID as int

AS

BEGIN

if CHARINDEX('.', @RecipientTableName) < 1

Begin

Set @RecipientTableName = @RecipientTableName + '.'

end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--This section is useful when debugging the Stored Proc

drop table #temp6

Declare @RecipientTableName AS nvarchar(250)

Declare @RecipientRowID as INT

Declare @LanguageID as int

Set @RecipientTableName = 'Environment.'

Set @RecipientRowID = 1

Set @LanguageID =50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--Step 1. Create a temp table to hold the field names for the passed in RecipientTableName and Language and rowID

CREATE TABLE #temp6 (

ID INT IDENTITY(1, 1)

,field nvarchar(250) )

--Step 2. Generate the SQL that will get the field names and save them to the temp table

DECLARE @sqlT AS varchar(max)

set @sqlT = 'Insert into #temp6(field) select LanguageKey from LanguageText where [Languagekey] like '''

+ @RecipientTableName

+ '%'' and RecipientID ='

+ Cast(@RecipientRowID as varchar(200))

+ ' and LanguageID = ' + Cast(@LanguageID as varchar(200)) + ''

Exec(@sqlT)

--Step 3. Get a count of the number of fields

DECLARE @FieldCount INT

select @FieldCount = count(ID)from #temp6

--Step 4. Declare and initialize variables

DECLARE @sql AS varchar(max)

SET @sql = ''

DECLARE @FieldName AS varchar(max)

SET @FieldName = ''

DECLARE @sql3 AS varchar(max)

SET @sql3 = ''

--Step 5. Loop through the temp table making the list of fields to be selected

WHILE @FieldCount > 0

BEGIN

set @FieldName = ''

select @FieldName = field from #temp6 where id = @FieldCount

SET @sql= @sql + ' ,Max( Case When LanguageKey = ''' + @FieldName + ''' then [Text] end) as [' + replace(@FieldName, @RecipientTableName, '') + ']'

set @FieldCount = @FieldCount - 1

End

--Step 6. Drop the temp table

drop table #temp6

--Step 7. Finish the SQL

DECLARE @FinishSQL AS varchar(max)

Set @FinishSQL = ' from (Select RecipientID, LanguageKey, [TEXT] from LanguageText '

Set @FinishSQL = @FinishSQL + ' where RecipientID =' + Cast(@RecipientRowID as varchar(200)) + ' and LanguageID = ' + Cast(@LanguageID as varchar(200)) + ' and [Languagekey] in( '

Set @FinishSQL = @FinishSQL + ' Select LanguageKey from LanguageText '

Set @FinishSQL = @FinishSQL + ' where [Languagekey] like ''' + @RecipientTableName + '%'' and RecipientID = ' + Cast(@RecipientRowID as varchar(200)) + ' and LanguageID = ' + Cast(@LanguageID as varchar(200)) + ')'

Set @FinishSQL = @FinishSQL + ') as TTT group by RecipientID'

--Step 8. Make and run the complete SQL

set @sql = 'select RecipientID' + @sql + @FinishSQL

exec(@sql)

END

/\*

The KEY to this proc is:

select RecipientID as EnviromentRowID

,Max( Case When LanguageKey = 'Environment.Description' then [Text] end) as [Description]

,Max( Case When LanguageKey = 'Environment.Test' then [Text] end) as [Test]

from

(

Select RecipientID, LanguageKey, [TEXT] from LanguageText where [Languagekey]

in( Select LanguageKey from LanguageText where [Languagekey] like 'Environment.%' and RecipientID = 1)

) as a

group by RecipientID

\*/

Here is an enum that matches the Language Table

public enum Language

{

Default,

af\_ZA, // Afrikaans - South Africa 0x0436 AFK

sq\_AL, // Albanian - Albania 0x041C SQI

ar\_DZ, // Arabic - Algeria 0x1401 ARG

ar\_BH, // Arabic - Bahrain 0x3C01 ARH

ar\_EG, // Arabic - Egypt 0x0C01 ARE

ar\_IQ, // Arabic - Iraq 0x0801 ARI

ar\_JO, // Arabic - Jordan 0x2C01 ARJ

ar\_KW, // Arabic - Kuwait 0x3401 ARK

ar\_LB, // Arabic - Lebanon 0x3001 ARB

ar\_LY, // Arabic - Libya 0x1001 ARL

ar\_MA, // Arabic - Morocco 0x1801 ARM

ar\_OM, // Arabic - Oman 0x2001 ARO

ar\_QA, // Arabic - Qatar 0x4001 ARQ

ar\_SA, // Arabic - Saudi Arabia 0x0401 ARA

ar\_SY, // Arabic - Syria 0x2801 ARS

ar\_TN, // Arabic - Tunisia 0x1C01 ART

ar\_AE, // Arabic - United Arab Emirates 0x3801 ARU

ar\_YE, // Arabic - Yemen 0x2401 ARY

hy\_AM, // Armenian - Armenia 0x042B

Cy\_az\_AZ, // Azeri (Cyrillic) - Azerbaijan 0x082C

Lt\_az\_AZ, // Azeri (Latin) - Azerbaijan 0x042C

eu\_ES, // Basque - Basque 0x042D EUQ

be\_BY, // Belarusian - Belarus 0x0423 BEL

bg\_BG, // Bulgarian - Bulgaria 0x0402 BGR

ca\_ES, // Catalan - Catalan 0x0403 CAT

zh\_CN, // Chinese - China 0x0804 CHS

zh\_HK, // Chinese - Hong Kong SAR 0x0C04 ZHH

zh\_MO, // Chinese - Macau SAR 0x1404

zh\_SG, // Chinese - Singapore 0x1004 ZHI

zh\_TW, // Chinese - Taiwan 0x0404 CHT

zh\_CHS, // Chinese (Simplified) 0x0004

zh\_CHT, // Chinese (Traditional) 0x7C04

hr\_HR, // Croatian - Croatia 0x041A HRV

cs\_CZ, // Czech - Czech Republic 0x0405 CSY

da\_DK, // Danish - Denmark 0x0406 DAN

div\_MV, // Dhivehi - Maldives 0x0465

nl\_BE, // Dutch - Belgium 0x0813 NLB

nl\_NL, // Dutch - The Netherlands 0x0413

en\_AU, // English - Australia 0x0C09 ENA

en\_BZ, // English - Belize 0x2809 ENL

en\_CA, // English - Canada 0x1009 ENC

en\_CB, // English - Caribbean 0x2409 ENB

en\_IE, // English - Ireland 0x1809 ENI

en\_JM, // English - Jamaica 0x2009 ENJ

en\_NZ, // English - New Zealand 0x1409 ENZ

en\_PH, // English - Philippines 0x3409 ENP

en\_ZA, // English - South Africa 0x1C09 ENS

en\_TT, // English - Trinidad and Tobago 0x2C09 ENT

en\_GB, // English - United Kingdom 0x0809 ENG

en\_US, // English - United States 0x0409 ENU

en\_ZW, // English - Zimbabwe 0x3009 ENW

et\_EE, // Estonian - Estonia 0x0425 ETI

fo\_FO, // Faroese - Faroe Islands 0x0438 FOS

fa\_IR, // Farsi - Iran 0x0429 FAR

fi\_FI, // Finnish - Finland 0x040B FIN

fr\_BE, // French - Belgium 0x080C FRB

fr\_CA, // French - Canada 0x0C0C FRC

fr\_FR, // French - France 0x040C FRA

fr\_LU, //French - Luxembourg 0x140C FRL

fr\_MC, // French - Monaco 0x180C FRM

fr\_CH, // French - Switzerland 0x100C FRS

gl\_ES, // Galician - Galician 0x0456

ka\_GE, // Georgian - Georgia 0x0437

de\_AT, // German - Austria 0x0C07 DEA

de\_DE, // German - Germany 0x0407 DEU

de\_LI, // German - Liechtenstein 0x1407 DEC

de\_LU, // German - Luxembourg 0x1007 DEL

de\_CH, // German - Switzerland 0x0807 DES

el\_GR, // Greek - Greece 0x0408 ELL

gu\_IN, // Gujarati - India 0x0447

he\_IL, // Hebrew - Israel 0x040D HEB

hi\_IN, // Hindi - India 0x0439 HIN

hu\_HU, // Hungarian - Hungary 0x040E HUN

is\_IS, // Icelandic - Iceland 0x040F ISL

id\_ID, // Indonesian - Indonesia 0x0421 IND

it\_IT, // Italian - Italy 0x0410

it\_CH, // Italian - Switzerland 0x0810 ITS

ja\_JP, // Japanese - Japan 0x0411 JPN (78)

kn\_IN, // Kannada - India 0x044B

kk\_KZ, // Kazakh - Kazakhstan 0x043F

kok\_IN, // Konkani - India 0x0457

ko\_KR, // Korean - Korea 0x0412 KOR (82)

ky\_KZ, // Kyrgyz - Kazakhstan 0x0440

lv\_LV, // Latvian - Latvia 0x0426 LVI

lt\_LT, // Lithuanian - Lithuania 0x0427 LTH

mk\_MK, // Macedonian (FYROM) 0x042F MKD

ms\_BN, // Malay - Brunei 0x083E MSB

ms\_MY, // Malay - Malaysia 0x043E MSL

mr\_IN, // Marathi - India 0x044E

mn\_MN, // Mongolian - Mongolia 0x0450

nb\_NO, // Norwegian (BokmÃ¥l) - Norway 0x0414 NOR

nn\_NO, // Norwegian (Nynorsk) - Norway 0x0814 NON

pl\_PL, // Polish - Poland 0x0415 PLK

pt\_BR, // Portuguese - Brazil 0x0416 PTB

pt\_PT, // Portuguese - Portugal 0x0816 PIG

pa\_IN, // Punjabi - India 0x0446

ro\_RO, // Romanian - Romania 0x0418 ROM

ru\_RU, // Russian - Russia 0x0419 RUS

sa\_IN, // Sanskrit - India 0x044F

Cy\_sr\_SP, // Serbian (Cyrillic) - Serbia 0x0C1A

Lt\_sr\_SP, // Serbian (Latin) - Serbia 0x081A

sk\_SK, // Slovak - Slovakia 0x041B SKY

sl\_SI, // Slovenian - Slovenia 0x0424 SLV

es\_AR, // Spanish - Argentina 0x2C0A ESS

es\_BO, // Spanish - Bolivia 0x400A ESB

es\_CL, // Spanish - Chile 0x340A ESL

es\_CO, // Spanish - Colombia 0x240A ESO

es\_CR, // Spanish - Costa Rica 0x140A ESC

es\_DO, // Spanish - Dominican Republic 0x1C0A ESD

es\_EC, // Spanish - Ecuador 0x300A ESF

es\_SV, // Spanish - El Salvador 0x440A ESE

es\_GT, // Spanish - Guatemala 0x100A ESG

es\_HN, // Spanish - Honduras 0x480A ESH

es\_MX, // Spanish - Mexico 0x080A ESM

es\_NI, // Spanish - Nicaragua 0x4C0A ESI

es\_PA, // Spanish - Panama 0x180A ESA

es\_PY, // Spanish - Paraguay 0x3C0A ESZ

es\_PE, // Spanish - Peru 0x280A ESR

es\_PR, // Spanish - Puerto Rico 0x500A ES

es\_ES, // Spanish - Spain 0x0C0A ESN

es\_UY, // Spanish - Uruguay 0x380A ESY

es\_VE, // Spanish - Venezuela 0x200A ESV

sw\_KE, // Swahili - Kenya 0x0441

sv\_FI, // Swedish - Finland 0x081D SVF

sv\_SE, // Swedish - Sweden 0x041D SVE

syr\_SY, // Syriac - Syria 0x045A

ta\_IN, // Tamil - India 0x0449

tt\_RU, // Tatar - Russia 0x0444

te\_IN, // Telugu - India 0x044A

th\_TH, // Thai - Thailand 0x041E THA

tr\_TR, // Turkish - Turkey 0x041F TRK

uk\_UA, // Ukrainian - Ukraine 0x0422 UKR

ur\_PK, // Urdu - Pakistan 0x0420 URD

Cy\_uz\_UZ, // Uzbek (Cyrillic) - Uzbekistan 0x0843

Lt\_uz\_UZ, // Uzbek (Latin) - Uzbekistan 0x0443

vi\_VN // Vietnamese - Vietnam 0x042A VIT

}