TECH-TPZ303-R

Full text search in Firebird without a full text search engine

Björn Reimer, Dirk Baumeister

Who we are?

Björn Reimer

- Working as DBA at the Friedrich-Alexander-Universität Erlangen Nürnberg
- Independent software developer

Dirk Baumeister

- Working as Computer Scientist at the Language Centre of the Friedrich-Alexander-Universität Erlangen Nürnberg
- Independent software developer

Requirements and intentions (1)

- Fast search in different smaller texts
- Consider stop words
- Multi language capable
- Access from windows applications (Delphi) and www with Perl or PHP
- Simple client apps
- No 3-tier application = Keep It Simple

Requirements and intentions (2)

- Only use “standard” UDFs
  (shipped with Firebird)
- Save meta information for texts
- Not too much time consumption when inserting a new text
- Compromise: medium performance is enough
- Reusable in other projects

Interface / API

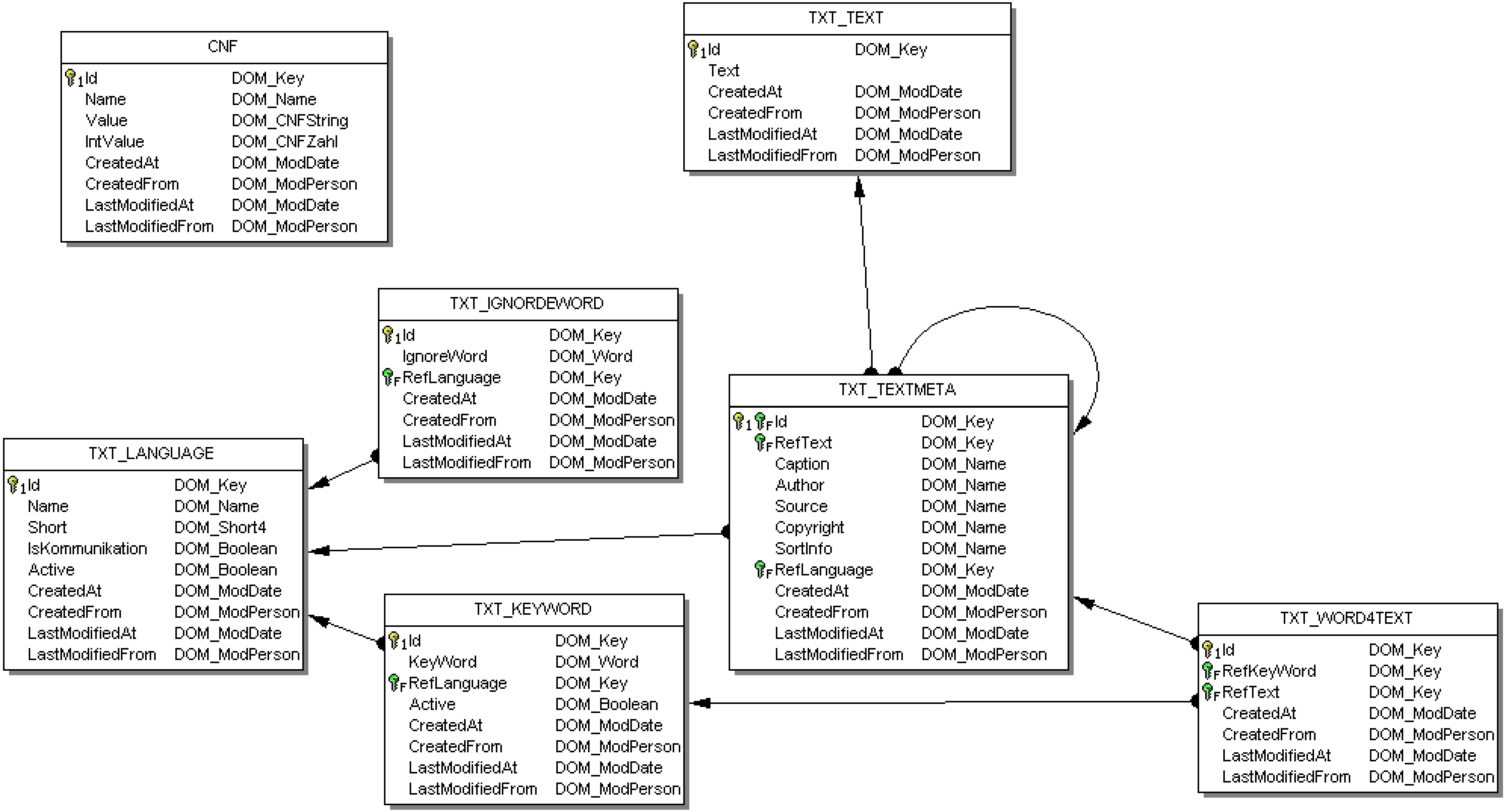
- Pure SQL with Stored Procedures
- Methods:
  - add text

```
PROCEDURE PRC_TXT_INSERT
("Text" VARCHAR(32000), "RefLanguage" BIGINT,
"RefPriorText" BIGINT)
RETURNS ("RefText" BIGINT)
```

  - find text:

```
PROCEDURE PRC_TXT_DOSEARCH ("Word" VARCHAR(80),
"RefLanguage" BIGINT)
RETURNS ("RefText" BIGINT)
```

Structure of DB



CNF

🔑1 Id	DOM_Key
Name	DOM_Name
Value	DOM_CNFString
IntValue	DOM_CNFZahl
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_TEXT

🔑1 Id	DOM_Key
Text	
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_IGNORDEWORD

🔑1 Id	DOM_Key
IgnoreWord	DOM_Word
🔑F RefLanguage	DOM_Key
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_TEXTMETA

🔑1 🔑F Id	DOM_Key
🔑F RefText	DOM_Key
Caption	DOM_Name
Author	DOM_Name
Source	DOM_Name
Copyright	DOM_Name
SortInfo	DOM_Name
🔑F RefLanguage	DOM_Key
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_LANGUAGE

🔑1 Id	DOM_Key
Name	DOM_Name
Short	DOM_Short4
IsKommunikation	DOM_Boolean
Active	DOM_Boolean
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_KEYWORD

🔑1 Id	DOM_Key
KeyWord	DOM_Word
🔑F RefLanguage	DOM_Key
Active	DOM_Boolean
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

TXT_WORD4TEXT

🔑1 Id	DOM_Key
🔑F RefKeyWord	DOM_Key
🔑F RefText	DOM_Key
CreatedAt	DOM_ModDate
CreatedFrom	DOM_ModPerson
LastModifiedAt	DOM_ModDate
LastModifiedFrom	DOM_ModPerson

Tables

- TXT_KEYWORD: keywords, which are already indexed
- TXT_IGNORDEWORD: keywords, which never will be indexed
- TXT_LANGUAGE: languages
- TXT_TEXT: the text
- TXT_TEXTMETA: meta info of text
- TXT_WORD4TEXT: found keywords in texts

Insert text

- Insert text into table TXT_TEXT
- If set in CNF: call PRC_TXT_UPDATEKEYWORDS for checking, whether known keywords are in the new text
- Otherwise call PRC_TXT_UPDATEALLKEYWORDS after bulk import
- Never automatically add new keywords to TXT_KEYWORD, when inserting texts

Find (part 1)

Use procedure PRC_TXT_DOSEARCH

- Check, if search word is a known keyword
- If not:
  - check, whether search word is in ignore word list
- If not:
  - Add search word as new keyword
  - Search in all texts for this word
  - Save result in TXT_WORD4TEXT

Find (part 2)

- If search word is a known keyword
  - search in TXT_WORD4TEXT

Always: If text is returned, check, whether text is root text otherwise follow links back to root text

In Client: Fetch all text information via separate SELECT.

Performance

- Good performance when inserting
- Poor performance, when searching keyword the first time
- Good performance, when searching keyword second time and so on

performance environment

- Environment
  Firebird 2.0 RC 5 (Super Server)
  Athlon 64 3800+ X2
  2 GB Ram
  Windows XP Pro
- Interface IBExpert with fbclient.dll on the same machine (executed at least twice via fetch all in sql window)
- about 210000 text records with always the same text (about 1 GB DB size)

performance for raw search

```
SELECT T."Id" FROM TXT_TEXT T
  WHERE T."Text" containing 'OKTIS';

PLAN (T NATURAL)
```

Prepare time = 0ms
Execute time = **34s 703ms**
Avg fetch time = 0,17 ms
Current memory = 969.996
Max memory = 1.054.760
Memory buffers = 2.048
Reads from disk to cache = 53.358
Writes from cache to disk = 0
Fetches from cache = 529.432

performance for already indexed word

```
SELECT T."Id" FROM TXT_TEXT T JOIN TXT_WORD4TEXT WT
  ON WT."RefText" = T."Id" WHERE WT."RefKeyWord"= 28;
```

```
PLAN JOIN (WT INDEX
  (FK_TXT_WORD4TEXT_RefKeyWord), T INDEX
  (PK_TXT_TEXT))
```

Prepare time = 16ms
Execute time = **375ms**
Avg fetch time = 0,04 ms
Current memory = 966.596
Max memory = 1.054.760
Memory buffers = 2.048
Reads from disk to cache = 0
Writes from cache to disk = 0
Fetches from cache = 70.015

performance for new word

```
SELECT "RefText"
  FROM PRC_TXT_DOSEARCH('OKTIS', 29);
```

Prepare time = 0ms
Execute time = **53s 750ms**
Avg fetch time = 0,26 ms
Current memory = 1.180.856
Max memory = 1.344.996
Memory buffers = 2.048
Reads from disk to cache = 56.633
Writes from cache to disk = 2.491
Fetches from cache = 7.083.702

performance of a known word

```
SELECT "RefText"
  FROM PRC_TXT_DOSEARCH('OKTIS', 29);
```

```
Prepare time = 0ms
  Execute time = 1s 828ms
  Avg fetch time = 0,01 ms
  Current memory = 1.106.748
  Max memory = 1.175.184
  Memory buffers = 2.048
  Reads from disk to cache = 0
  Writes from cache to disk = 0
  Fetches from cache = 420.090
```

application range

Easily usable in existing dbs

- Search in free text
- Search in xml data
- ...

enhancements

- Use BLOB instead of Varchar
  (more complex in client)
- Linguistic enhancements (e. g. case)
- Parser for import
  - XML
  - HTML
- More intelligent search methods

The END

- Contact for questions and improvements:

  Björn Reimer (reimer@softbaer.de)

  Dirk Baumeister (baumeister@softbaer.de)