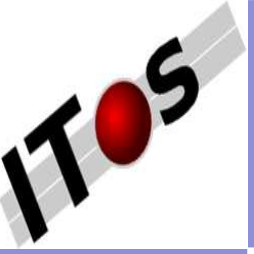


Ortsuche

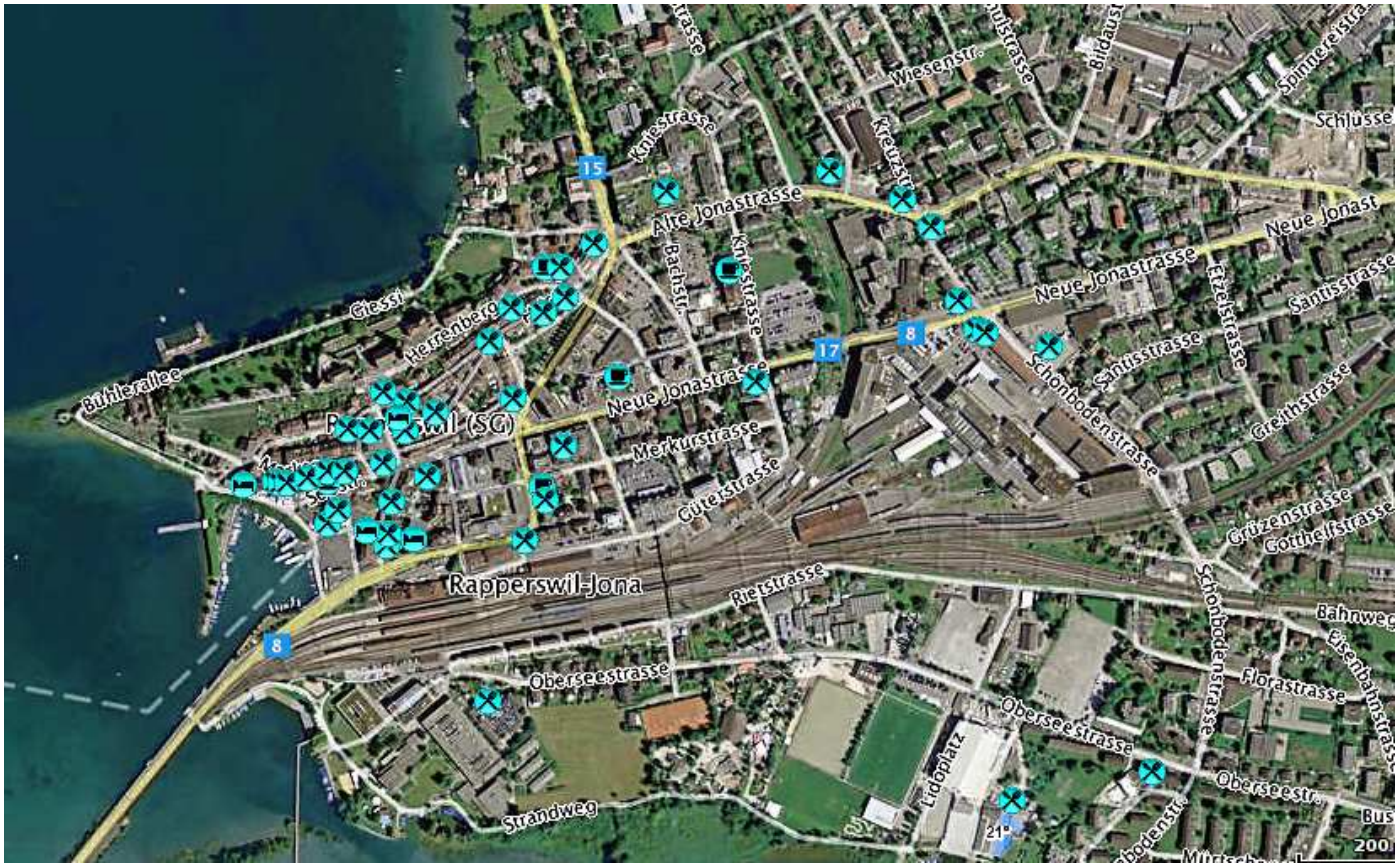
mit
PostgreSQL und PostGIS

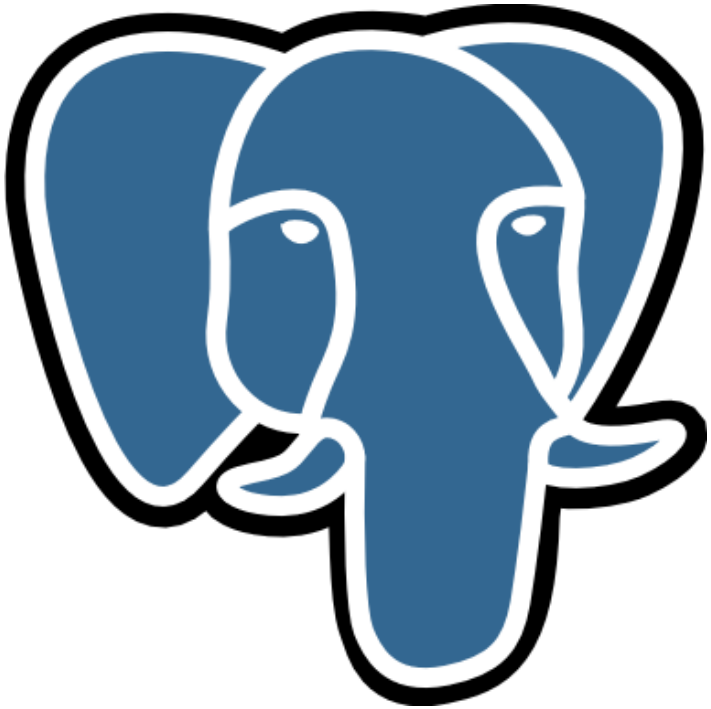
Stephan Wagner

ITos GmbH, CH-9642 Ebnat-Kappel

FOSSGIS
13. Juni 2013

Ziel





Hinweise an den *Mahaut*

- Ortskoordinaten
  - Koordinatensystem
  - Einheiten
- Ortsname
  - Sprache
  - Schreibweise
  - Rechtschreibung
  - Wie hei[ss,ß]t das Ding schon wieder?

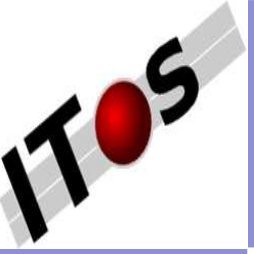
- Ortskoordinaten
  - Koordinatensystem
  - Einheiten
- Ortsname
  - Sprache
  - Schreibweise
  - Rechtschreibung
  - Wie hei[ss,ß]t das Ding schon wieder?

Preambel
Beispiel Ortsbezeichnung

- Geographisch: (PostgreSQL oder PostGIS-Geometrietypen)
  - Koordinate Ortszentrum (**Punkt**)
  - Umhüllende des Ortes (Linie)
  - Ortsfläche
- Sprachlich:
  - Fribourg (französisch)
  - Frybùrg (alemannisch)
  - Friboua, Fribôrg (franko-provenzalisch)
  - Freiburg, Freiburg Ü(e)chtland
  - Freiburg, Freiburg im Breisgau, Friburg im Brisgau (alemannisch)
- Stichworte: Nähe, Radius, Abstand, Nearest Neighbour, kNN-Gist-Index $\Rightarrow$ Nach Abstand sortiert

Preambel
Beispiel Ortsbezeichnung

- Geographisch: (PostgreSQL oder PostGIS-Geometrietypen)
  - Koordinate Ortszentrum (**Punkt**)
  - Umhüllende des Ortes (Linie)
  - Ortsfläche
- Sprachlich:
  - Fribourg (französisch)
  - Frybùrg (alemannisch)
  - Friboua, Fribôrg (franko-provenzalisch)
  - Freiburg, Freiburg Ü(e)chtland
  - Freiburg, Freiburg im Breisgau, Friburg im Brisgau (alemannisch)
- Stichworte: Nähe, Radius, Abstand, Nearest Neighbour, kNN-Gist-Index $\Rightarrow$ Nach Abstand sortiert

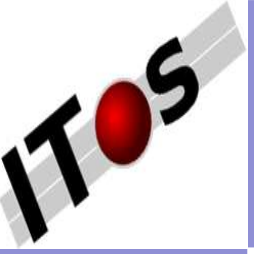


Preambel
Beispiel Ortsbezeichnung

- Geographisch: (PostgreSQL oder PostGIS-Geometrietypen)
  - Koordinate Ortszentrum (**Punkt**)
  - Umhüllende des Ortes (Linie)
  - Ortsfläche
- Sprachlich:
  - Fribourg (französisch)
  - Frybùrg (alemannisch)
  - Friboua, Fribôrg (franko-provenzalisch)
  - Freiburg, Freiburg Ü(e)chtland
  - Freiburg, Freiburg im Breisgau, Friburg im Brisgau (alemannisch)
- Stichworte: Nähe, Radius, Abstand, Nearest Neighbour, kNN-Gist-Index $\Rightarrow$ Nach Abstand sortiert

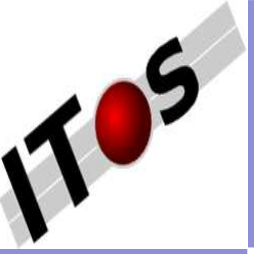
Preambel
Hinweise

1. Spracheinstellung und gewähltes Wörterbuch beeinflussen Suchergebnis
2. Einbettung (PL/pgSQL-)Funktionen
3. Autocomplete



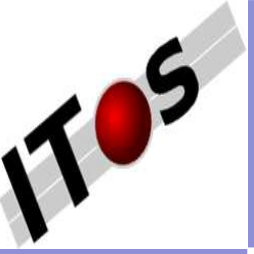
Preambel
Hinweise

1. Spracheinstellung und gewähltes Wörterbuch beeinflussen Suchergebnis
2. Einbettung (PL/pgSQL-)Funktionen
3. Autocomplete



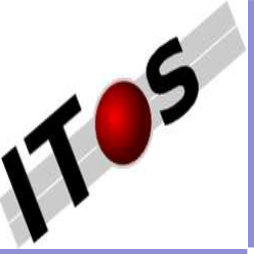
Preambel
Hinweise

1. Spracheinstellung und gewähltes Wörterbuch beeinflussen Suchergebnis
2. Einbettung (PL/pgSQL-)Funktionen
3. Autocomplete



Preambel
PostgreSQL-Erweiterungen einbinden

- `ls /usr/share/postgresql/9.2/extension` oder `SELECT * FROM pg_available_extensions` listet bereits installierte Erweiterungen.
- `CREATE EXTENSION 'ExtensionName' WITH SCHEMA 'Schema' VERSION 'Version' FROM 'AlteVersion'`
- **Debian:** aptitude install postgresql-contrib-9.2
- Ansonsten kompilieren (besp. PostGIS 2.x)



Preambel
Versionen

- PostgreSQL 9.2
- PostGIS 2.0.3
- pg_trgm 1.0
- fuzzystrmatch 1.0
- unaccent 1.0

Preambel
Referent

- Stephan Wagner
- Geschäftsführer *ITos GmbH*
- Web-(GIS)-Lösungen
- PostgreSQL - Django (SQL, Python, HTML, CSS, JS)
- ⇒ System-Architektur und -Administration

Preambel
Referent

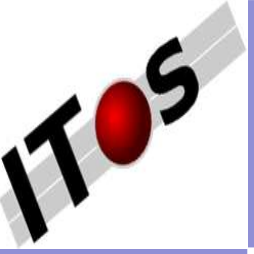
- Stephan Wagner
- Geschäftsführer *ITos GmbH*
- Web-(GIS)-Lösungen
- PostgreSQL - Django (SQL, Python, HTML, CSS, JS)
- ⇒ System-Architektur und -Administration

Preambel
Referent

- Stephan Wagner
- Geschäftsführer *ITos GmbH*
- Web-(GIS)-Lösungen
- PostgreSQL - Django (SQL, Python, HTML, CSS, JS)
- $\Rightarrow$ System-Architektur und -Administration

Themen

Inhaltsverzeichnis



SQLpostgres
Suche nach Spalteninhalt

- `SELECT * FROM myTable;`
- `SELECT row2, row1 * 1.5 AS row1new FROM myTable;`

- `SELECT row1 FROM myTable WHERE row2 = 'OrtsName';`
- `SELECT row1 FROM myTable WHERE row2 > 'OrtsName1' AND row2 < 'OrtsName2';`
- `SELECT row2 FROM myTable WHERE row1 < (SELECT ABS(AVG(row1) - row1) FROM myTable;`

- Auf Suchspalten INDEX legen
- Ab PostgreSQL 9.1 GiST kNN-INDEX
- Ab PostgreSQL 9.3 SELECT nur auf INDEX möglich
- **Beachte:**
  - Planer-Ausgabe analysieren (EXPLAIN ANALYSE)
  - impliziter SORT
  - RAM für INDEX, SORT (PG tuning)

SQLpostgres
Suche Beschleunigen: INDEX

- Auf Suchspalten INDEX legen
- Ab PostgreSQL 9.1 GiST kNN-INDEX
- Ab PostgreSQL 9.3 SELECT nur auf INDEX möglich
- **Beachte:**
  - Planer-Ausgabe analysieren (EXPLAIN ANALYSE)
  - impliziter SORT
  - RAM für INDEX, SORT (PG tuning)

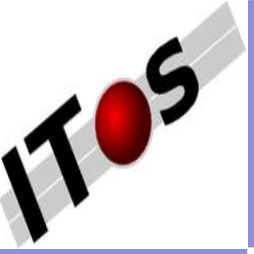
SQLpostgres
Suche Beschleunigen: INDEX

- Auf Suchspalten INDEX legen
- Ab PostgreSQL 9.1 GiST kNN-INDEX
- Ab PostgreSQL 9.3 SELECT nur auf INDEX möglich
- Beachte:
  - Planer-Ausgabe analysieren (EXPLAIN ANALYSE)
  - impliziter SORT
  - RAM für INDEX, SORT (PG tuning)

SQLpostgres
Suche Beschleunigen: INDEX

- Auf Suchspalten INDEX legen
- Ab PostgreSQL 9.1 GiST kNN-INDEX
- Ab PostgreSQL 9.3 SELECT nur auf INDEX möglich
- **Beachte:**
  - Planer-Ausgabe analysieren (EXPLAIN ANALYSE)
  - impliziter SORT
  - RAM für INDEX, SORT (PG tuning)

- Auf Suchspalten INDEX legen
- Ab PostgreSQL 9.1 GiST kNN-INDEX
- Ab PostgreSQL 9.3 SELECT nur auf INDEX möglich
- **Beachte:**
  - Planer-Ausgabe analysieren (EXPLAIN ANALYSE)
  - impliziter SORT
  - RAM für INDEX, SORT (PG tuning)



SQLpostgres
Einschränkende Suche: CASE WHEN


```
SELECT CASE WHEN row1='OrtNameFR' THEN 'OrtNameDE'
            WHEN row2='OrtNameDE' THEN 'OrtNameDE'
            ELSE NONE
       END
  FROM myTable;
```

Mustersuche auf Begriff:

- `SELECT * FROM myTable WHERE row2 LIKE '%Ortsna%';`
- `SELECT row1 FROM myTable WHERE row2 ILIKE '%rtsNa%';`
- `SELECT row1 FROM myTable WHERE lower(row2) LIKE '%rtsna%';`

Mustersuche auf ganze Zeichenkette:

- `SELECT * FROM myTable WHERE row1 SIMLAR TO 'Ortsna%';`
- `SELECT * FROM myTable WHERE row2 SIMLAR TO '(Ort|ort)%';`
- `SELECT * FROM myTable WHERE row2 SIMLAR TO '[O,o]rt%';`

Vergleichsoperator	Beschreibung
!	Negation
$\sim$	keysensitiver Vergleich
$\sim *$	key**in**sensitiver Vergleich

Suche in etwa 'Karlstrasse', Strasse in der Schreibweise ß:

- `SELECT * FROM myTable WHERE row2 '^[C,c,K,k]arl+[ß]+$';`

SQLpostgres

PostgreSQL

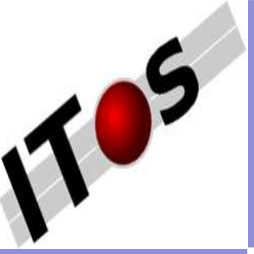
- Koordinatensystem
- Einheiten
- **PostgreSQL:**
  - Punkt (*point*), Gerade (*lseg*), Linie (*path*), Polygon (*polygon*), Rechteck (*box*), Kreis (*circle*)
- **PostGIS:**
  - GiST-kNN-Index mit <-> als Distanzoperator
  - `SELECT ort FROM ortsnamen ORDER BY geom <-> st_setsrid(st_makepoint(-90,40),4326) LIMIT 10;`
  - ⇒ Die 10 zum Testpunkt nächstgelegenen Punkte

- Koordinatensystem
- Einheiten
- **PostgreSQL:**
  - Punkt (*point*), Gerade (*lseg*), Linie (*path*), Polygon (*polygon*), Rechteck (*box*), Kreis (*circle*)
- **PostGIS:**
  - GiST-kNN-Index mit <-> als Distanzoperator
  - `SELECT ort FROM ortsnamen ORDER BY geom <-> st_setsrid(st_makepoint(-90,40),4326) LIMIT 10;`
  - $\Rightarrow$ Die 10 zum Testpunkt nächstgelegenen Punkte

- Extension aus dem Umfeld von FTS
- Datei mit Konvertierungsregeln unter `$SHAREDIR/tsearch_data/`
- Filter, welcher Lexeme ($\sim$ Begriffe) von Akzenten befreit
- **Abfrage:** `SELECT unaccent('Freiburg im Üechtland');`
  $\Rightarrow$ Freiburg im Uechtland

unaccent
unaccent

- Extension aus dem Umfeld von FTS
- Datei mit Konvertierungsregeln unter `$SHAREDIR/tsearch_data/`
- Filter, welcher Lexeme ($\sim$ Begriffe) von Akzenten befreit
- **Abfrage:** `SELECT unaccent('Freiburg im Üechtland');`
  $\Rightarrow$ Freiburg im Uechtland



FuzzyStrMatch
PG-Extension

- Soundex
- (Double-)Metaphone
- Levenshtein

FuzzyStrMatch
Soundex

- **Algorithmus:** Robert Russell und Margaret Odel, 1918
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Ergibt auch für Deutsch und Französich brauchbare Ergebnisse
- **Code:** Erster Buchstabe plus drei Ziffern, bsp. W-213 für Wikipedia.
  - **Wort** mit mehr Buchstaben als benötigt ⇒ Schnitt
  - **Wort** mit wenig Buchstaben als benötigt ⇒ Nullen
- `SELECT SOUNDEX('Fribourg')` ⇒ F616
- 'Fryburg' ⇒ F616 ⇐ 'Freiburg im Üechtland'
- `SELECT * FROM ort WHERE difference(ort.nm, 'Friburg') > 2;`
- Britney Spears ⇒ B635 S162 ⇐ Bewährten Superzicke ☺

FuzzyStrMatch
Soundex

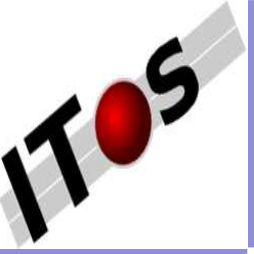
- **Algorithmus:** Robert Russell und Margaret Odel, 1918
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Ergibt auch für Deutsch und Französich brauchbare Ergebnisse
- **Code:** Erster Buchstabe plus drei Ziffern, bsp. W-213 für Wikipedia.
  - **Wort** mit mehr Buchstaben als benötigt ⇒ Schnitt
  - **Wort** mit wenig Buchstaben als benötigt ⇒ Nullen
- `SELECT SOUNDEX('Fribourg')` ⇒ F616
- 'Fryburg' ⇒ F616 ⇐ 'Freiburg im Üechtland'
- `SELECT * FROM ort WHERE difference(ort.nm, 'Friburg') > 2;`
- Britney Spears ⇒ B635 S162 ⇐ Bewährten Superzicke ☺

FuzzyStrMatch
Soundex

- **Algorithmus:** Robert Russell und Margaret Odel, 1918
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Ergibt auch für Deutsch und Französich brauchbare Ergebnisse
- **Code:** Erster Buchstabe plus drei Ziffern, bsp. W-213 für Wikipedia.
  - **Wort** mit mehr Buchstaben als benötigt ⇒ Schnitt
  - **Wort** mit wenig Buchstaben als benötigt ⇒ Nullen
- `SELECT SOUNDEX('Fribourg')` ⇒ F616
- 'Fryburg' ⇒ F616 ⇐ 'Freiburg im Üechtland'
- `SELECT * FROM ort WHERE difference(ort.nm, 'Friburg') > 2;`
- Britney Spears ⇒ B635 S162 ⇐ Bewährten Superzicke ☺

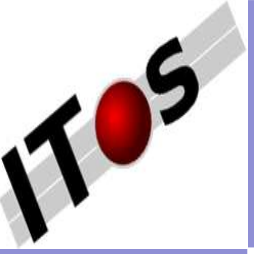
FuzzyStrMatch
Soundex

- **Algorithmus:** Robert Russell und Margaret Odel, 1918
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Ergibt auch für Deutsch und Französich brauchbare Ergebnisse
- **Code:** Erster Buchstabe plus drei Ziffern, bsp. W-213 für Wikipedia.
  - **Wort** mit mehr Buchstaben als benötigt ⇒ Schnitt
  - **Wort** mit wenig Buchstaben als benötigt ⇒ Nullen
- `SELECT SOUNDEX('Fribourg')` ⇒ F616
- 'Fryburg' ⇒ F616 ⇐ 'Freiburg im Üechtland'
- `SELECT * FROM ort WHERE difference(ort.nm, 'Friburg') > 2;`
- Britney Spears ⇒ B635 S162 ⇐ Bewährten Superzicke ☺



FuzzyStrMatch
(Double-)Mataphone

- **Algorithmus:** Lawrence Phillips, 2000
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Absichtlich phonetisch unscharfe Darstellung (Sprechweisen)
- Unterscheidet feingranularer als SOUNDEX
- Gewichtung der Operationen möglich
- `SELECT metaphone('Fryburg im Üechtland', 7);` ⇒ FRBRKMX"
- `SELECT dmetaphone('Fryburg im Üechtland');` ⇒ FRPR"



FuzzyStrMatch
(Double-)Mataphone

- **Algorithmus:** Lawrence Phillips, 2000
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Absichtlich phonetisch unscharfe Darstellung (Sprechweisen)
- Unterscheidet feingranularer als SOUNDEX
- Gewichtung der Operationen möglich
- `SELECT metaphone('Fryburg im Üechtland', 7);` ⇒ FRBRKMX"
- `SELECT dmetaphone('Fryburg im Üechtland');` ⇒ FRPR"

FuzzyStrMatch
(Double-)Mataphone

- **Algorithmus:** Lawrence Phillips, 2000
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Absichtlich phonetisch unscharfe Darstellung (Sprechweisen)
- Unterscheidet feingranularer als SOUNDEX
- Gewichtung der Operationen möglich
- `SELECT metaphone('Fryburg im Üechtland', 7);` ⇒ FRBRKMX"
- `SELECT dmetaphone('Fryburg im Üechtland');` ⇒ FRPR"

FuzzyStrMatch
(Double-)Mataphone

- **Algorithmus:** Lawrence Phillips, 2000
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Absichtlich phonetisch unscharfe Darstellung (Sprechweisen)
- Unterscheidet feingranularer als SOUNDEX
- Gewichtung der Operationen möglich
- `SELECT metaphone('Fryburg im Üechtland', 7);` ⇒ FRBRKMX"
- `SELECT dmetaphone('Fryburg im Üechtland');` ⇒ FRPR"

FuzzyStrMatch
(Double-)Mataphone

- **Algorithmus:** Lawrence Phillips, 2000
- Phonetische Indizierung nach englischsprachigem Klangmuster
- Absichtlich phonetisch unscharfe Darstellung (Sprechweisen)
- Unterscheidet feingranularer als SOUNDEX
- Gewichtung der Operationen möglich
- `SELECT metaphone('Fryburg im Üechtland', 7);` ⇒ FRBRKMX"
- `SELECT dmetaphone('Fryburg im Üechtland');` ⇒ FRPR"

FuzzyStrMatch
Levenshtein Distanz

- **Algorithmus:** Wladimir Levenshtein, 1965
- **Distanz:** Anzahl von Einzeloperationen, um Zeichenkette 1 auf 2 umzuwandeln
- **Haupteinsatzbereich:** Duplikatserkennung, Rechtschreibeprüfung
- Bei grössen Datensätzen langsam: *O(NxM)*
- `SELECT levenshtein('Fribourg', 'Freibourg');` ⇒ 1
- `SELECT levenshtein('Fribourg', 'Fryburg im Üechtland');` ⇒ 15

FuzzyStrMatch
Levenshtein Distanz

- **Algorithmus:** Wladimir Levenshtein, 1965
- **Distanz:** Anzahl von Einzeloperationen, um Zeichenkette 1 auf 2 umzuwandeln
- **Haupteinsatzbereich:** Duplikatserkennung, Rechtschreibeprüfung
- Bei grössen Datensätzen langsam: *O(NxM)*
- `SELECT levenshtein('Fribourg', 'Freibourg');` ⇒ 1
- `SELECT levenshtein('Fribourg', 'Fryburg im Üechtland');` ⇒ 15

FuzzyStrMatch
Levenshtein Distanz

- **Algorithmus:** Wladimir Levenshtein, 1965
- **Distanz:** Anzahl von Einzeloperationen, um Zeichenkette 1 auf 2 umzuwandeln
- **Haupteinsatzbereich:** Duplikatserkennung, Rechtschreibeprüfung
- Bei grössen Datensätzen langsam: *O(NxM)*
- `SELECT levenshtein('Fribourg', 'Freibourg');` ⇒ 1
- `SELECT levenshtein('Fribourg', 'Fryburg im Üechtland');` ⇒ 15

FuzzyStrMatch
Levenshtein Distanz

- **Algorithmus:** Wladimir Levenshtein, 1965
- **Distanz:** Anzahl von Einzeloperationen, um Zeichenkette 1 auf 2 umzuwandeln
- **Haupteinsatzbereich:** Duplikatserkennung, Rechtschreibeprüfung
- Bei grössen Datensätzen langsam: *O(NxM)*
- `SELECT levenshtein('Fribourg', 'Freibourg');` ⇒ 1
- `SELECT levenshtein('Fribourg', 'Fryburg im Üechtland');` ⇒ 15

FuzzyStrMatch
Levenshtein Distanz

- **Algorithmus:** Wladimir Levenshtein, 1965
- **Distanz:** Anzahl von Einzeloperationen, um Zeichenkette 1 auf 2 umzuwandeln
- **Haupteinsatzbereich:** Duplikatserkennung, Rechtschreibeprüfung
- Bei grössen Datensätzen langsam: *O(NxM)*
- `SELECT levenshtein('Fribourg', 'Freibourg');` ⇒ 1
- `SELECT levenshtein('Fribourg', 'Fryburg im Üechtland');` ⇒ 15

Trigramm
Begriffe

- **Module:** pg_trgm
- **Trigramm:** Zerlegung eines Textes in 3-Zeichen-Fragmente (n-Gramm)
- **Trigramme:** Zur Weissagung dienende, altchinesiche Symbole
- **Dice-Koeffizient** d: Anteil übereinstimmender N-Gramme zweier Terme ($0 \leq d \leq 1$)

Trigramm
Begriffe

- **Module:** pg_trgm
- **Trigramm:** Zerlegung eines Textes in 3-Zeichen-Fragmente (n-Gramm)
- **Trigramme:** Zur Weissagung dienende, altchinesiche Symbole
- **Dice-Koeffizient** d**:** Anteil übereinstimmender N-Gramme zweier Terme ($0 \leq d \leq 1$)

- $d(a,b) = \frac{2|T(a) \cap T(b)|}{|T(a)|+|T(b)|}$
- **Beispiel:**
  - Term a = "wirk", Term b = "work"
  - $T(a) = \S\S w, \S wi, wir, irk, rk\S, k\S\S$
  - $T(b) = \S\S w, \S wo, wor, ork, rk\S, k\S\S$
  - $T(a) \cap T(b) = \S\S w, k\S\S, rk\S$
  - $d(wirk, work) = \frac{2 \cdot 3}{6+6} = \frac{1}{2}$

    d.h., Die Ähnlichkeit oder Distanz beträgt 0.5

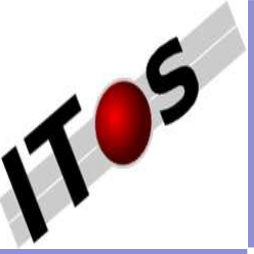
- Seit PostgreSQL 8.3 fester Bestandteil
- Resultatmenge: *Dokumente*, welche mit dem Suchbegriff möglichst nah Übereinstimmen
- Basis-Funktionalität:
  1. *Tokens* auf *Lexeme* reduziert, Stopwords gefiltert: *Bréil sur Royas* $\Rightarrow$ *breil roya* (Datentyp *tsvector* oder *tsvector)*
  2. *GIN* oder *kNN-GiST*-Index auf Spalte Lexem
  3. Suchanfrage mit to_tsquery() und dem Matching-Operator &&
  4. Ausgabe des Suchergebnisses nach Relevanz

- Seit PostgreSQL 8.3 fester Bestandteil
- Resultatmenge: *Dokumente*, welche mit dem Suchbegriff möglichst nah Übereinstimmen
- Basis-Funktionalität:
  1. *Tokens* auf *Lexeme* reduziert, Stopwords gefiltert: *Bréil sur Royas* $\Rightarrow$ *breil roya* (Datentyp *tsvector* oder *tsvector)*
  2. *GIN* oder *kNN-GiST*-Index auf Spalte Lexem
  3. Suchanfrage mit to_tsquery() und dem Matching-Operator &&
  4. Ausgabe des Suchergebnisses nach Relevanz

- Seit PostgreSQL 8.3 fester Bestandteil
- Resultatmenge: *Dokumente*, welche mit dem Suchbegriff möglichst nah Übereinstimmen
- Basis-Funktionalität:
  1. *Tokens* auf *Lexeme* reduziert, Stopwords gefiltert: *Bréil sur Royas* $\Rightarrow$ *breil roya* (Datentyp *tsvector* oder *tsvector)*
  2. *GIN* oder *kNN-GiST*-Index auf Spalte Lexem
  3. Suchanfrage mit to_tsquery() und dem Matching-Operator &&
  4. Ausgabe des Suchergebnisses nach Relevanz

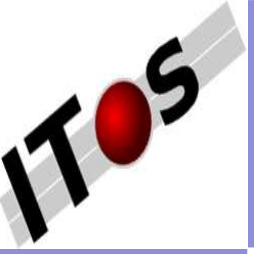
- Seit PostgreSQL 8.3 fester Bestandteil
- Resultatmenge: *Dokumente*, welche mit dem Suchbegriff möglichst nah Übereinstimmen
- Basis-Funktionalität:
  1. *Tokens* auf *Lexeme* reduziert, Stopwords gefiltert: *Bréil sur Royas* $\Rightarrow$ *breil roya* (Datentyp *tsvector* oder *tsvector)*
  2. *GIN* oder *kNN-GiST*-Index auf Spalte Lexem
  3. Suchanfrage mit to_tsquery() und dem Matching-Operator &&
  4. Ausgabe des Suchergebnisses nach Relevanz

- Seit PostgreSQL 8.3 fester Bestandteil
- Resultatmenge: *Dokumente*, welche mit dem Suchbegriff möglichst nah Übereinstimmen
- Basis-Funktionalität:
  1. *Tokens* auf *Lexeme* reduziert, Stopwords gefiltert: *Bréil sur Royas* $\Rightarrow$ *breil roya* (Datentyp *tsvector* oder *tsvector)*
  2. *GIN* oder *kNN-GiST*-Index auf Spalte Lexem
  3. Suchanfrage mit to_tsquery() und dem Matching-Operator &&
  4. Ausgabe des Suchergebnisses nach Relevanz



FTS
Sprachspezifische Wörterbücher

- Simple, **Synomym**, **Thesaurus**, Ispell, **Snowball** (Stemming-Algorithmus)
- *ls /usr/share/postgresql/9.2/tsearch_data*
- `CREATE TEXT SEARCH DISCTIONARY ...`
- **Synonym:** Ur-Begriff ⊔ Synonym
- **Thesaurus:** Ur-Satz : Synonym-Satz



FTS
Sprachspezifische Wörterbücher

- Simple, **Synomym**, **Thesaurus**, Ispell, **Snowball** (Stemming-Algorithmus)
- *ls /usr/share/postgresql/9.2/tsearch_data*
- `CREATE TEXT SEARCH DISCTIONARY ...`
- **Synonym:** Ur-Begriff ⊔ Synonym
- **Thesaurus:** Ur-Satz : Synonym-Satz

- Simple, **Synomym**, **Thesaurus**, Ispell, **Snowball** (Stemming-Algorithmus)
- *ls /usr/share/postgresql/9.2/tsearch_data*
- `CREATE TEXT SEARCH DISCTIONARY ...`
- **Synonym:** Ur-Begriff ⊔ Synonym
- **Thesaurus:** Ur-Satz : Synonym-Satz

Fazit

1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent

Fazit

1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent

Fazit

1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent

Fazit

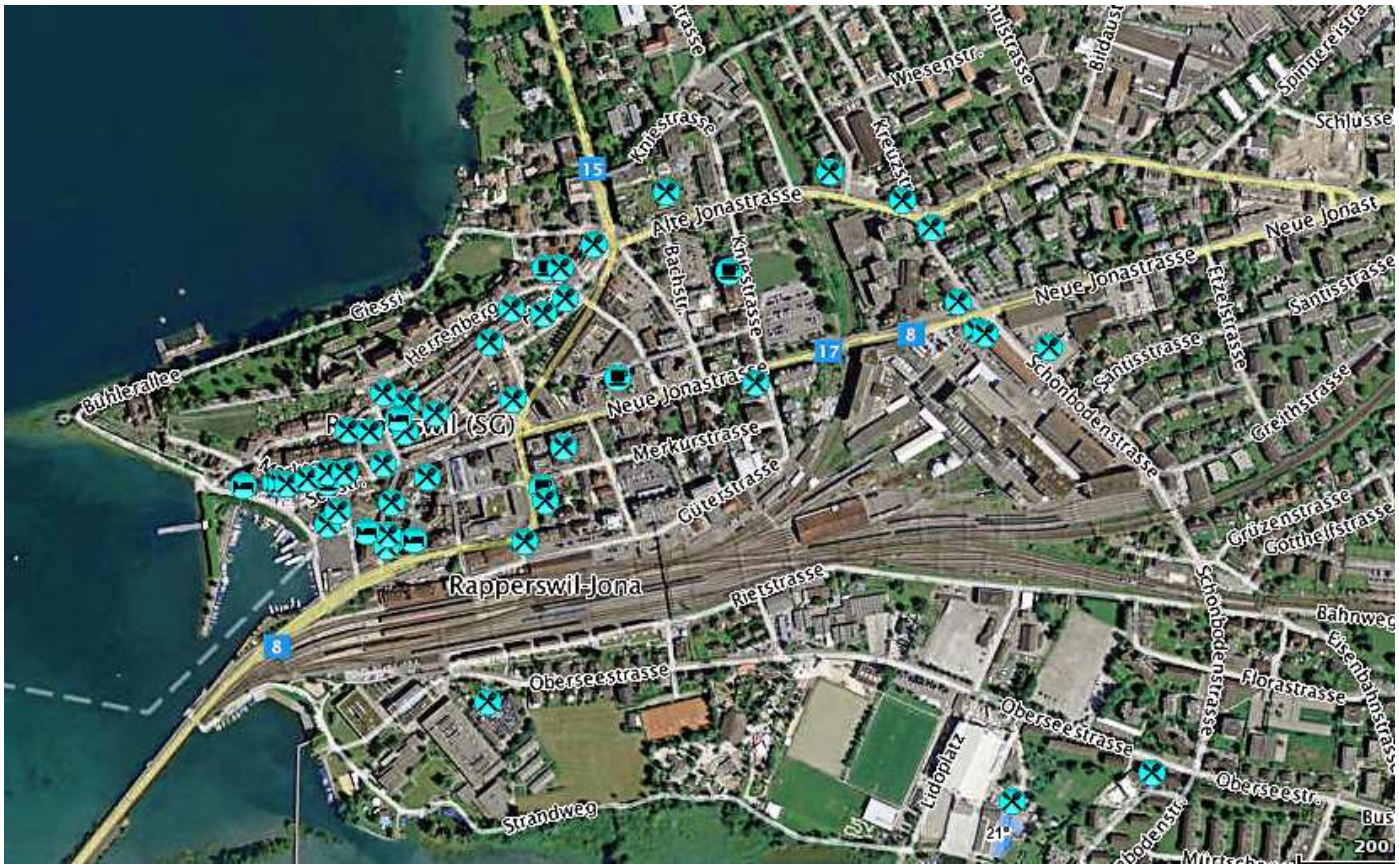
1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent

Fazit

1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent

Fazit

1. *Distanz* verbindet Ortsnamensuche und Ortskoordinatensuche
2. Ortsnamensuche mit Standard-SQL quasi statisch
3. PostgreSQL bietet Alternativen:
   - Soundex, Metaphone
   - Levenshtein
   - Trigramm
   - FTS
   - unaccent



oder

FOSSGIS e.V. Sektempfang