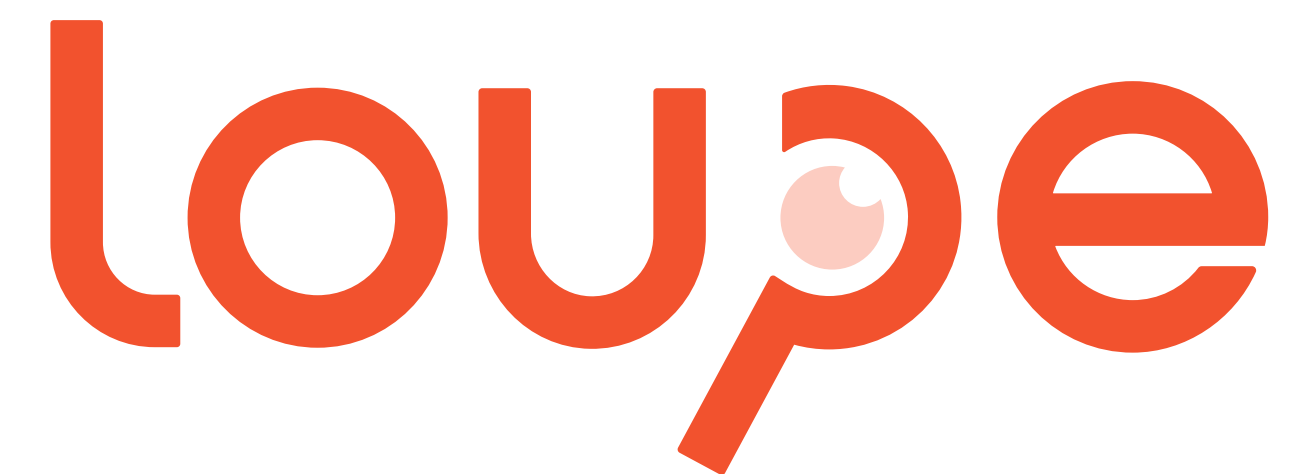


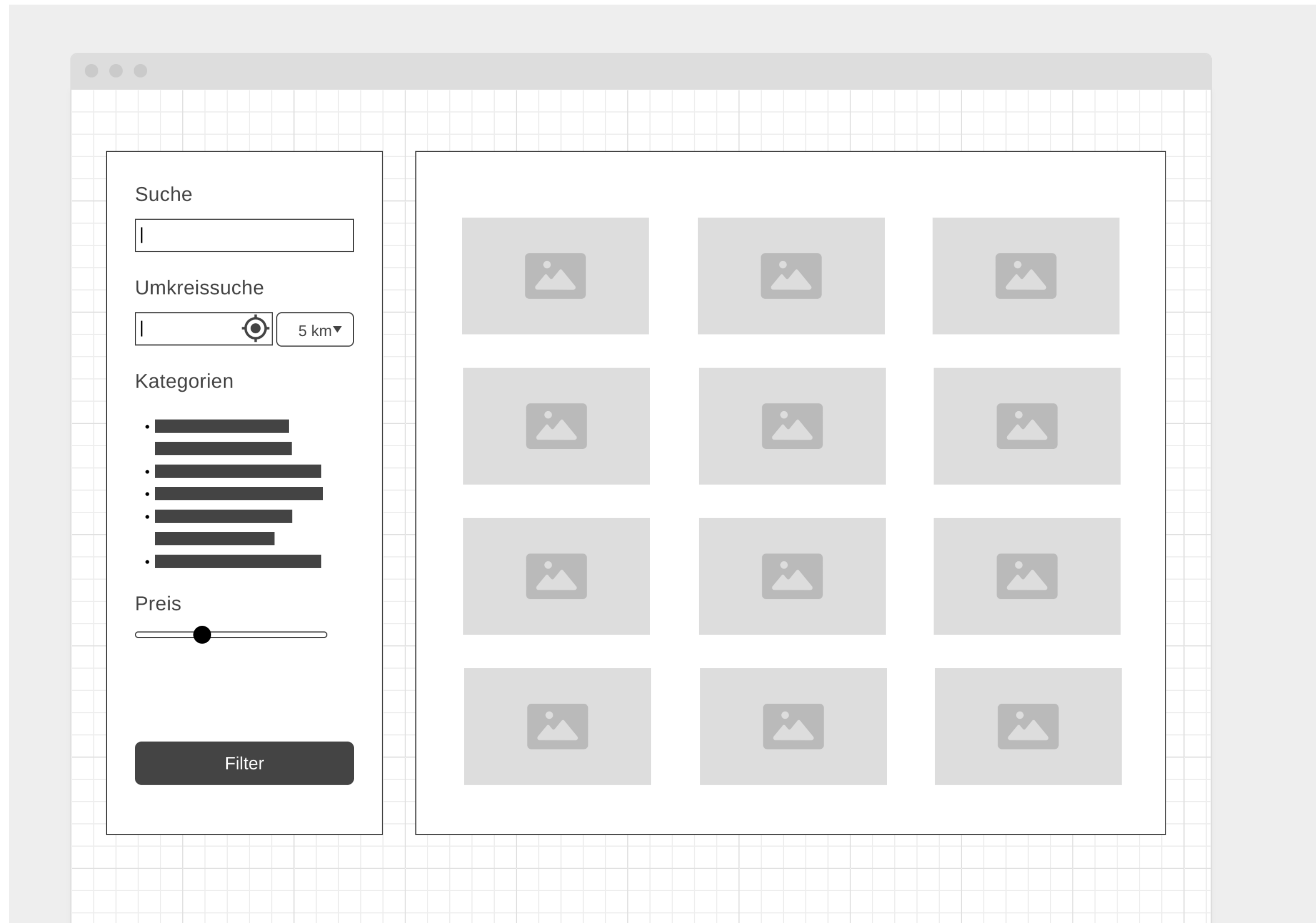
Loupe - wie schreibt
man eine Suchmaschine
nur mit **SQLite** und **PHP**?



Meine heutige Geschichte

- 2022
- Grosser Kunde im Tourismus-Bereich
- Neue Anforderung, typische POI-Listen
 - Unterkünfte
 - Wanderstrecken
 - Restaurants

Mockup



Anforderung: Tippfehler

Wenn ich
«Grindlewald»
tippe, soll
«Grindelwald»
gefunden werden.



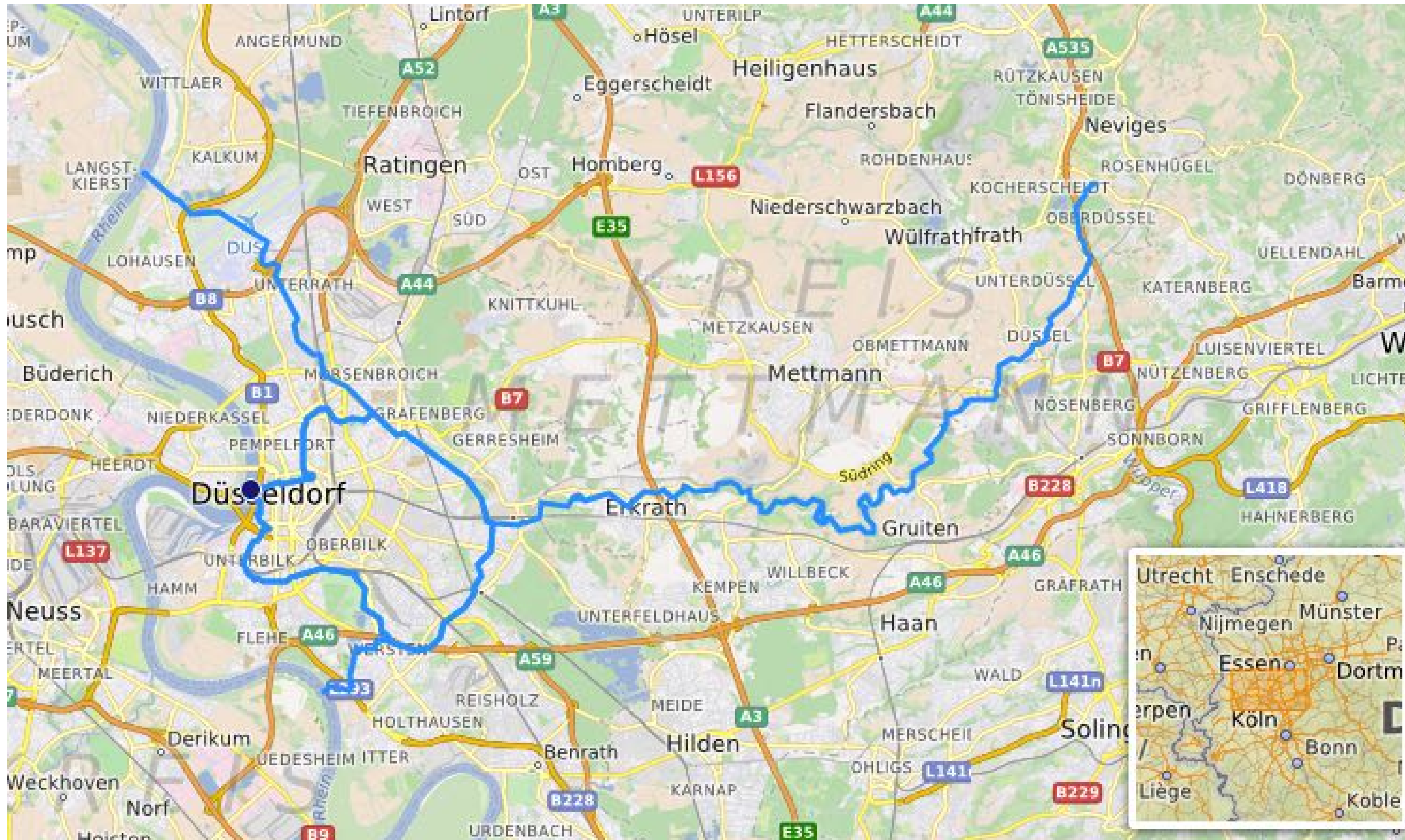
«handle»

«bundle»

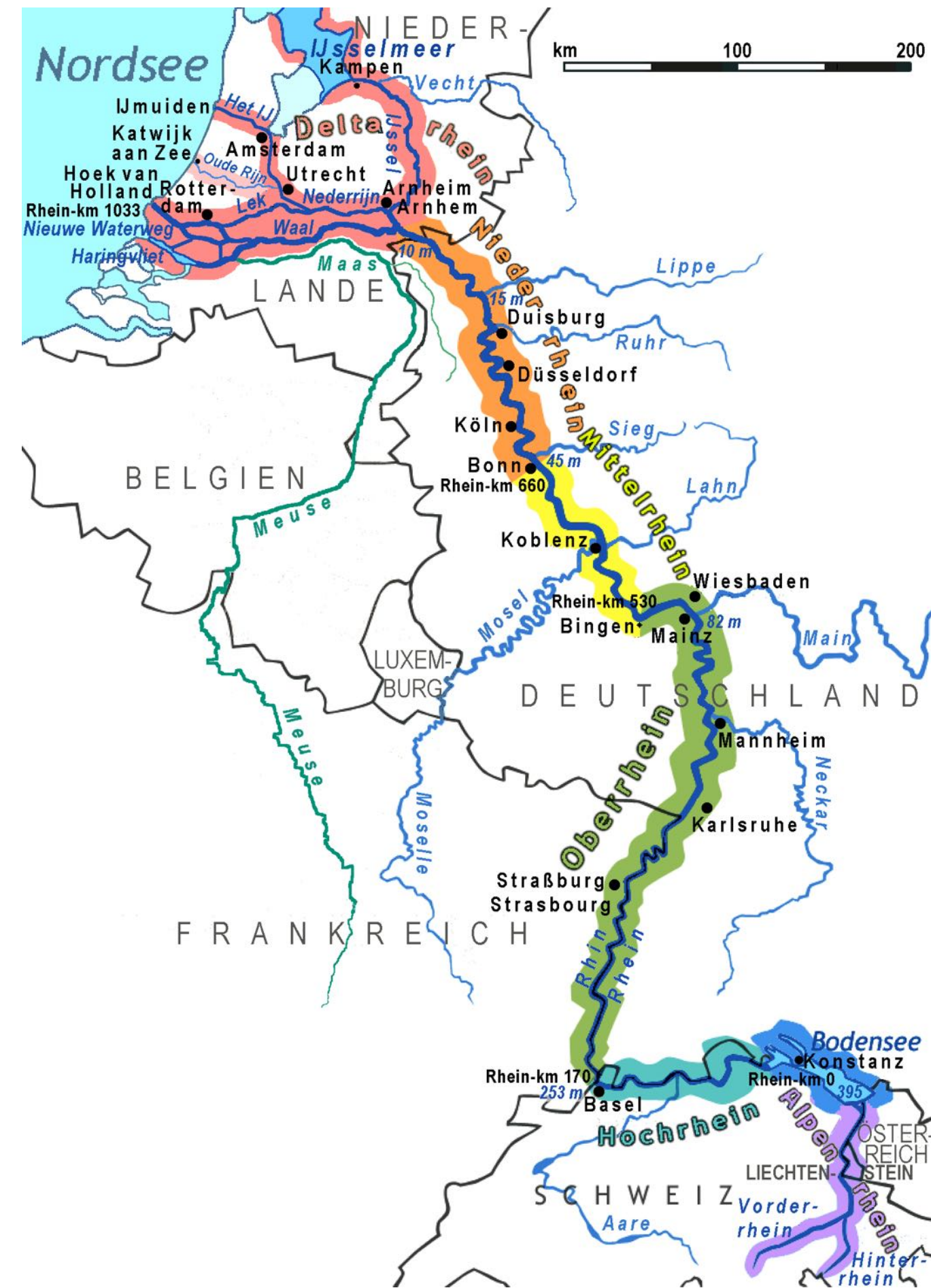
«candle»



Geografischer Exkurs



Der Rhein



Die Aare

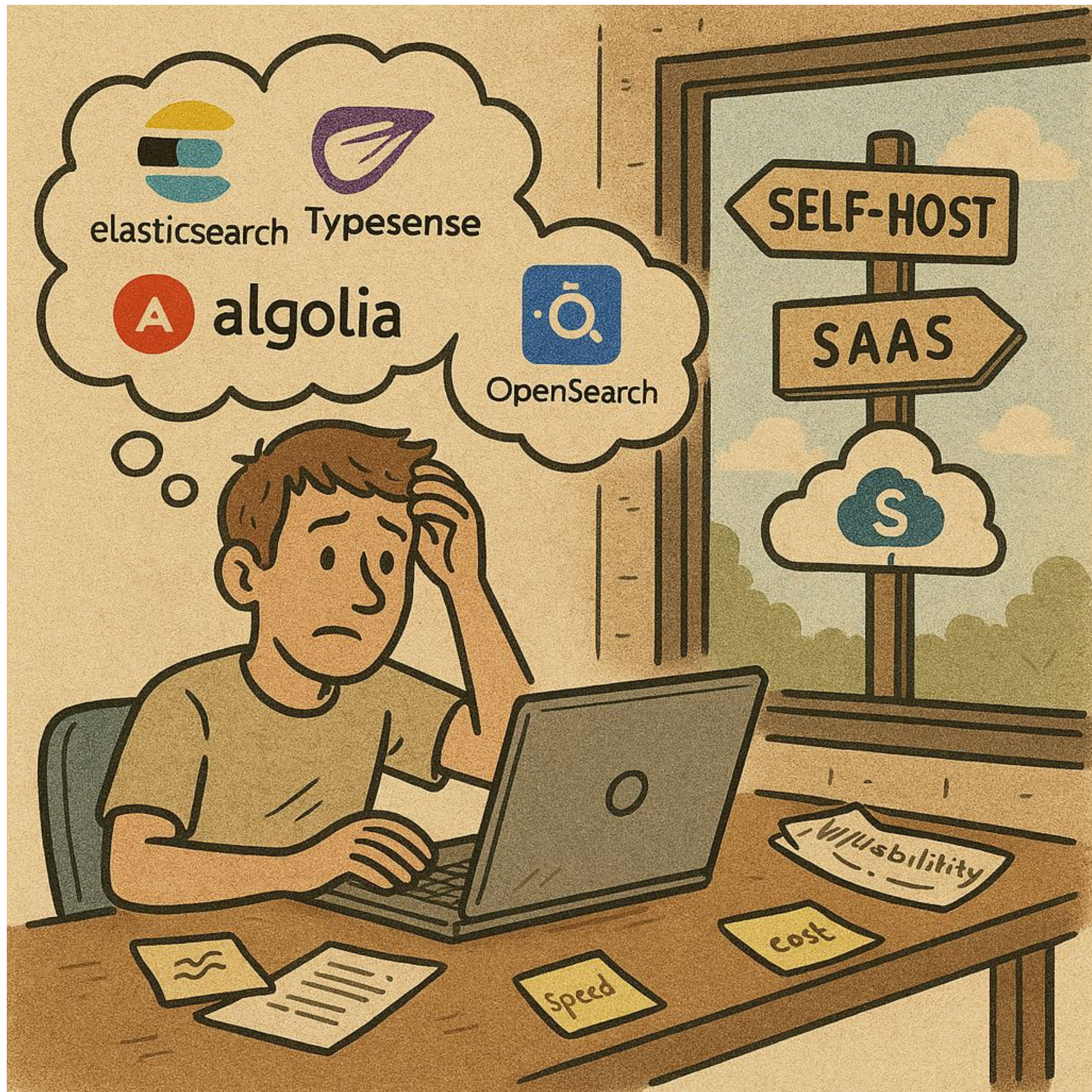


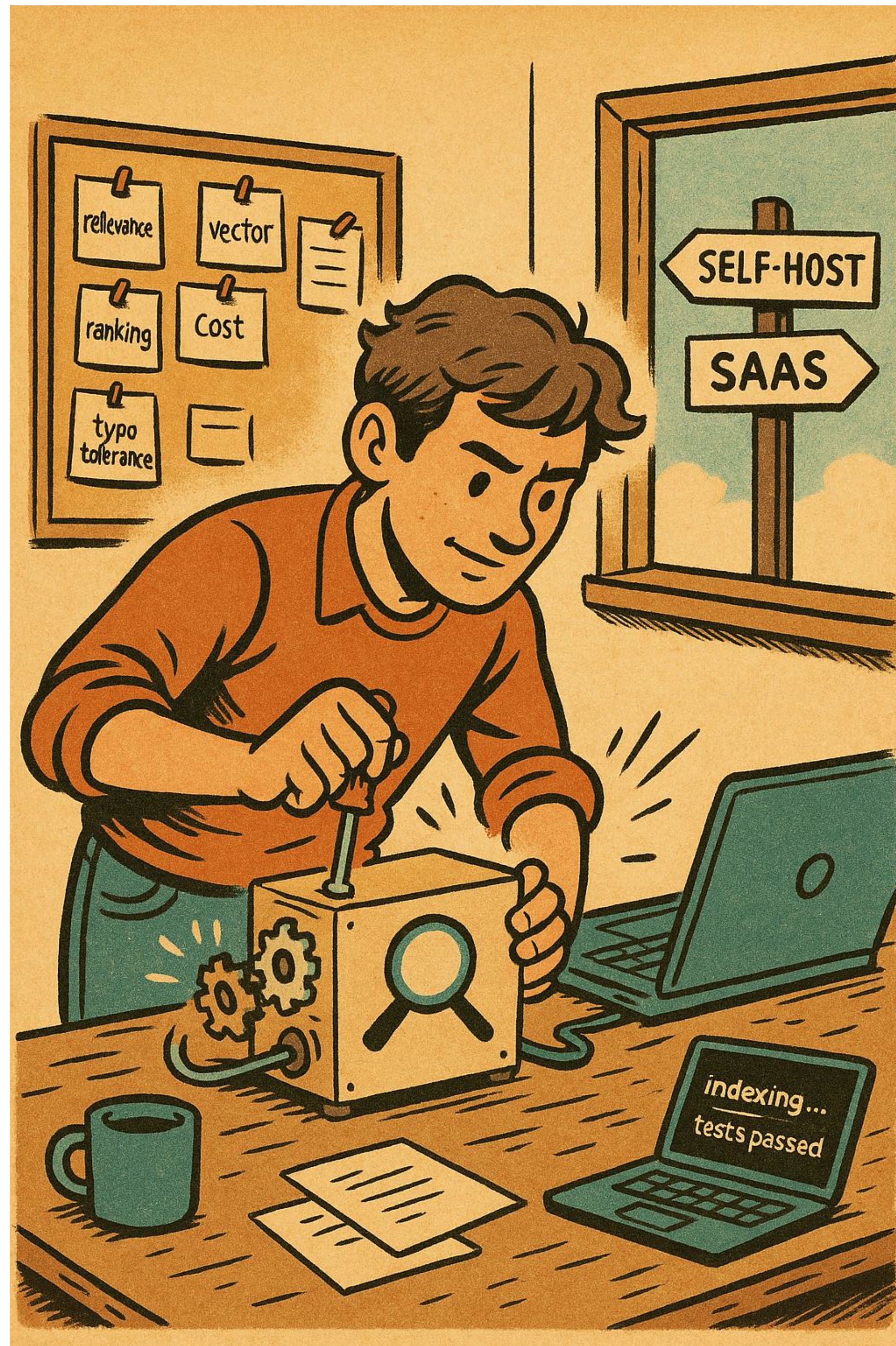
Bern



Grindelwald









Pdo\Sqlite::createFunction

(PHP 8 >= 8.4.0)

Pdo\Sqlite::createFunction — Registers a user-defined function for use in SQL statements

Examples

Example #1 Pdo\Sqlite::createFunction() example

In this example, we have a function that calculates the SHA256 sum of a string, and then reverses it. It returns the value of the filename transformed by our function. The data returned in `$rows` contains

The beauty of this technique is that there is no need to process the result using a [foreach](#) loop after

```
<?php
function sha256_and_reverse($string)
{
    return strrev(hash('sha256', $string));
}

$db = new Pdo\Sqlite('sqlite::sqlitedb');
$db->sqliteCreateFunction('sha256rev', 'sha256_and_reverse', 1);
$rows = $db->query('SELECT sha256rev(filename) FROM files')->fetchAll();
```

<?php

```
$db = new PDO('sqlite::memory:');
```

```
function loupe_levenshtein(string $string1, string $string2): int
{
    return levenshtein($string1, $string2);
}
```

```
$db->query("CREATE TABLE terms (id INTEGER PRIMARY KEY AUTOINCREMENT, term TEXT UNIQUE);")
$db->query("INSERT INTO terms (term) VALUES ('Grindelwald')");
```

```
$db->sqliteCreateFunction('loupe_levenshtein', 'loupe_levenshtein', 2);
```

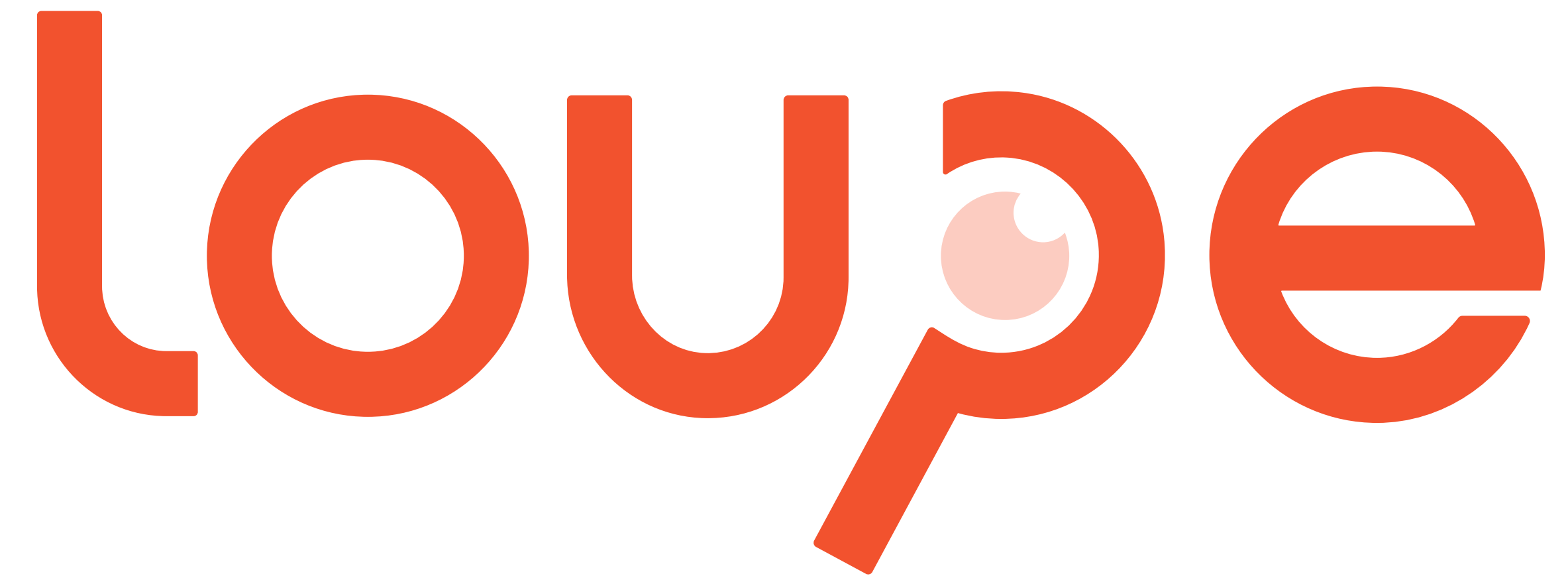
```
$stmt = $db->query("SELECT id, loupe_levenshtein('Grindelwald', term) AS typos FROM terms");
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
print_r($rows);
```

```
/**
 * Array
 * (
 *     [0] => Array
 *     (
 *         [id] => 1
 *         [typos] => 2
 *     )
 * )
 */
```

Anforderungen

- **Plattform**-Anforderungen:
 - **PHP** und **SQLite**
- **DX**-Anforderungen:
 - Einfache API (nicht so wie bei Elasticsearch) -> **Meilisearch** als Vorbild
- **Funktionale**-Anforderungen:
 - **Filter**-Logik (Kategorien, Preis, geografische Distanz)
 - **Sortierung** nach Alphabet, geografischer Distanz und natürlich nach Relevanz
 - **Typo-Tolerance** für einfache Buchstabendreher



1. Commit im Januar 2023

Document

{

"id": "rest 001",

"name": "Grindelwald Panorama Stube",

"teaser": "Gemütliche Berghütte mit atemberaubendem Blick auf die Nordwand des Eigers

"categories": ["Schweizer Küche", "Alpin", "Fondue"],

"stars": 5,

"average_price": 3,

"coordinates": { "lat": 46.6242, "lng": 8.0414 }

}

DX - API



```
$configuration = Configuration::create()  
->withSearchableAttributes(['name', 'teaser'])  
->withFilterableAttributes(['categories', 'stars', 'average_price', 'coordinates'])  
->withSortableAttributes(['average_price'])  
->withLanguages(['de'])  
;
```



```
$searchParameters = SearchParameters::create()  
->withQuery('grindlewald')  
->withFilter('stars >= 3')  
->withSort(['average_price:asc'])  
;
```

Grundlagen

| documents | | | |
|-----------|----------|--|--|
| id | user_id | document | |
| 1 | rest_001 | {"id":"rest_001","name":"Grindelwald Panorama S... | |
| 2 | rest_002 | {"id":"rest_002","name":"Lauterbrunnen Falls K\... | |
| 3 | rest_003 | {"id":"rest_003","name":"Gstaad Alpin Bistro","... | |
| 4 | rest_004 | {"id":"rest_004","name":"Kandersteg Chaletstube... | |
| 5 | rest_005 | {"id":"rest_005","name":"Interlaken Seehaus","t... | |
| | | | |
| | | | |

| documents | |
|----------------|--|
| key | value |
| documentSchema | {"id":"string","name":"string","teaser":"string","categories":"array<string>","stars": |
| engineVersion | 0.12.0 |
| configHash | 1bb1f87f742ca289d68d506b5458a39ae6116a81f69e8fed4a1c10cd34008491 |
| | |
| | |

Filtern

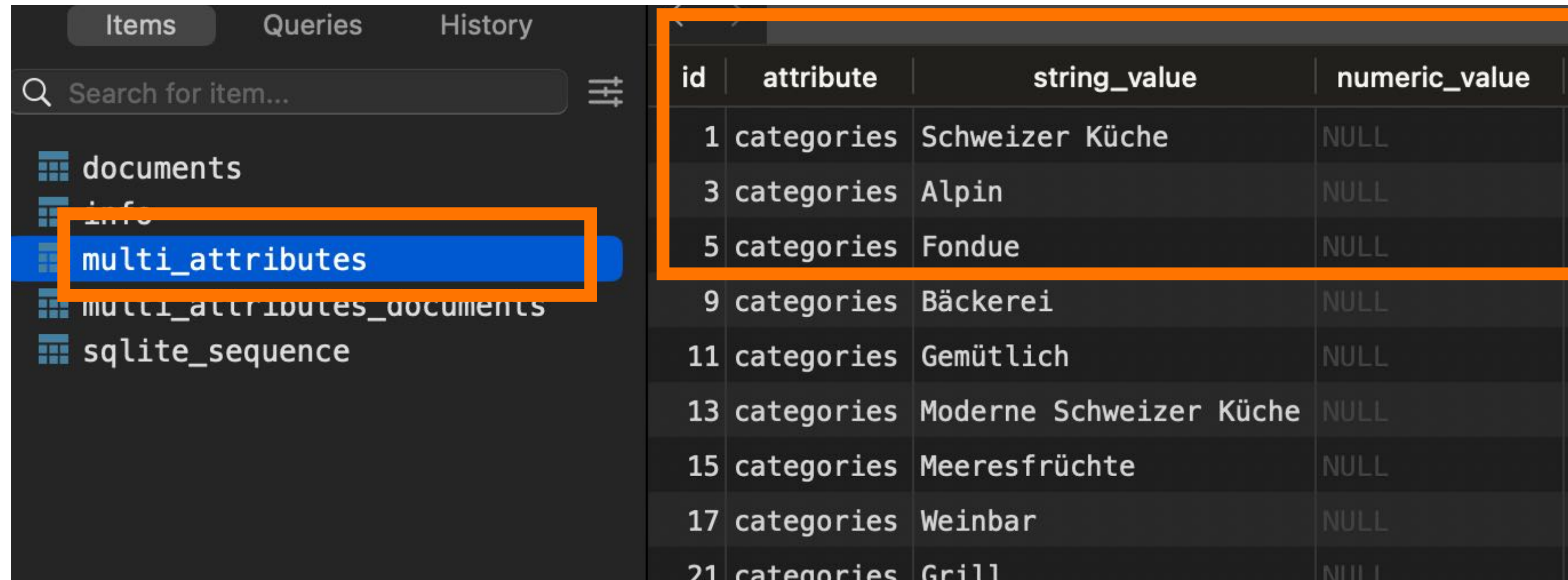
- **Single Attributes:** `stars, average_price`
- **Multi Attributes:** `categories`

Single Attribute

| documents | | | | | |
|-----------|----------|---|---------------|-------|----------------|
| id | user_id | document | average_price | stars | coordinates_ge |
| 1 | rest_001 | {"id":"rest_001","name":"Grindelwald Panor... | 3.0 | 5.0 | 46.6242 |
| 3 | rest_002 | {"id":"rest_002","name":"Lauterbrunnen Fal... | 2.0 | 4.0 | 46.5939 |
| 5 | rest_003 | {"id":"rest_003","name":"Gstaad Alpin Bist... | 3.0 | 4.0 | 46.4725 |
| 7 | rest_004 | {"id":"rest_004","name":"Kandersteg Chalet... | 2.0 | 3.0 | 46.494 |
| 9 | rest_005 | {"id":"rest_005","name":"Interlaken Seehau... | 1.0 | 2.0 | 46.6863 |

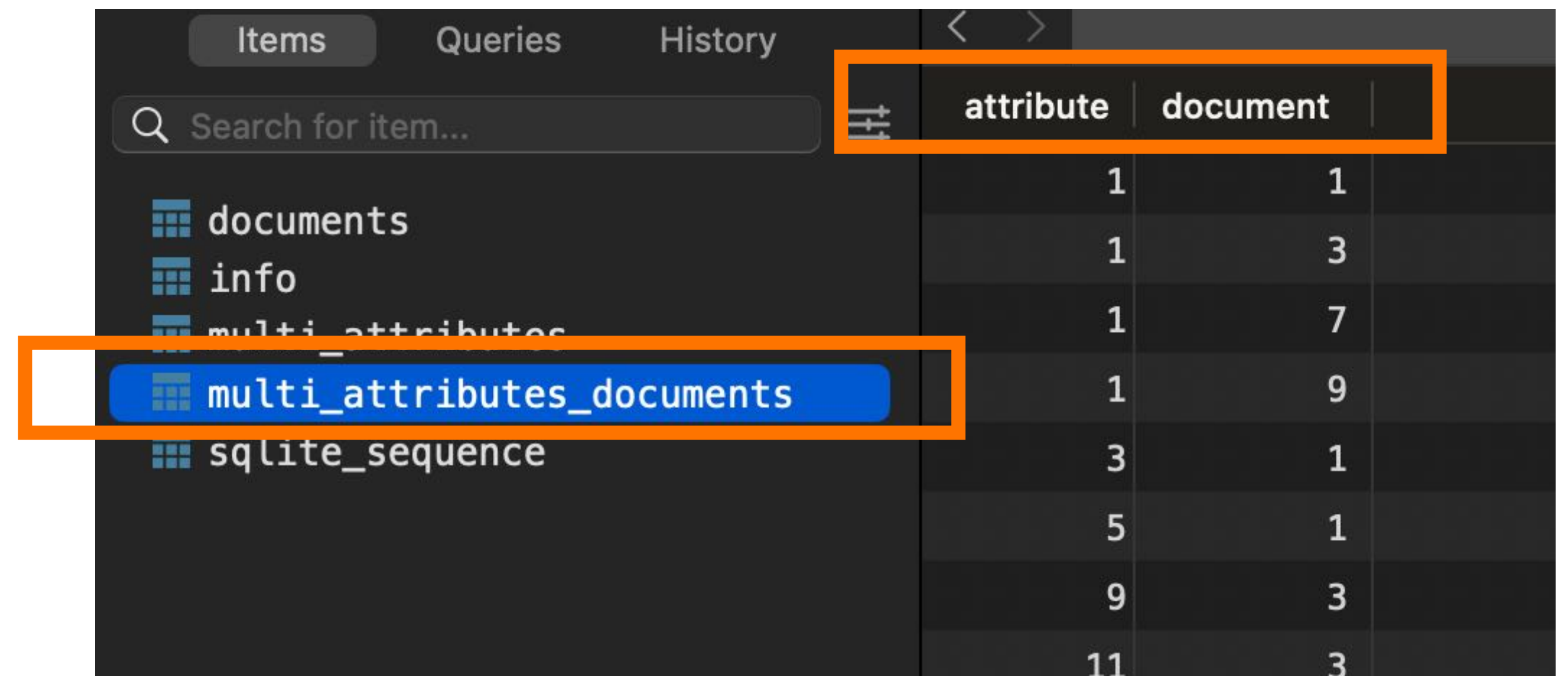
```
SELECT * FROM documents WHERE stars >= 3.0
```

Multi Attribute



The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of items: documents, info, multi_attributes (highlighted with an orange box), multi_attributes_documents, and sqlite_sequence. The main panel displays a table with the following data:

| id | attribute | string_value | numeric_value |
|----|------------|-------------------------|---------------|
| 1 | categories | Schweizer Küche | NULL |
| 3 | categories | Alpin | NULL |
| 5 | categories | Fondue | NULL |
| 9 | categories | Bäckerei | NULL |
| 11 | categories | Gemütlich | NULL |
| 13 | categories | Moderne Schweizer Küche | NULL |
| 15 | categories | Meeresfrüchte | NULL |
| 17 | categories | Weinbar | NULL |
| 21 | categories | Grill | NULL |



The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of items: documents, info, multi_attributes, multi_attributes_documents (highlighted with an orange box), and sqlite_sequence. The main panel displays a table with the following data:

| id | attribute | document |
|----|-----------|----------|
| 1 | 1 | 1 |
| 1 | 1 | 3 |
| 1 | 1 | 7 |
| 1 | 1 | 9 |
| 3 | 3 | 1 |
| 5 | 5 | 1 |
| 9 | 9 | 3 |
| 11 | 11 | 3 |

Multi Attribute

```
SELECT
  *
FROM
  documents d
WHERE
  d.id IN (
    SELECT
      *
    FROM
      multi_attributes_documents mad
      INNER JOIN multi_attributes ma ON ma.attribute = 'categories'
      AND ma.id = mad.attribute
      INNER JOIN documents d ON d.id = mad.document
    WHERE
      string_value = 'Fondue' OR string_value = 'Alpin'
  )
```

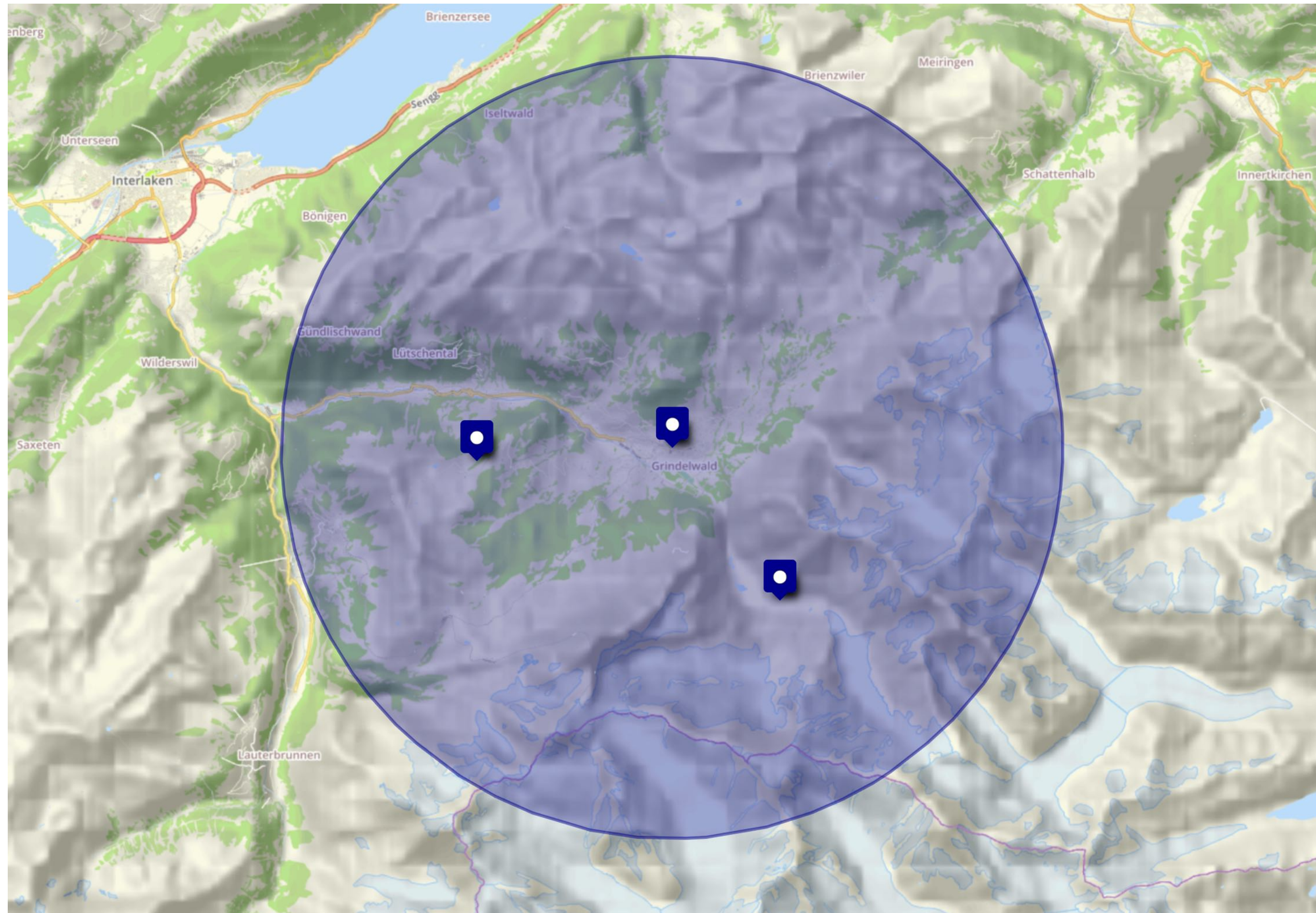
Kombination - Parser

```
$searchParameters = SearchParameters::create()  
->withFilter("((stars BETWEEN 3 AND 4) OR (categories = 'Alpin' OR categories = 'Fondue'))")  
;  
  
$result = $loupe->search($searchParameters);
```

- Lexer / Parser
- doctrine/lexer
- Am Ende ein längerer Query aber **SQLite** ist schnell



Filtern: Geo-Distanz



Filtern: Geo-Distanz

History

Filtern: Geo-Distanz

WHERE

loupe_geo_distance

46.625829,

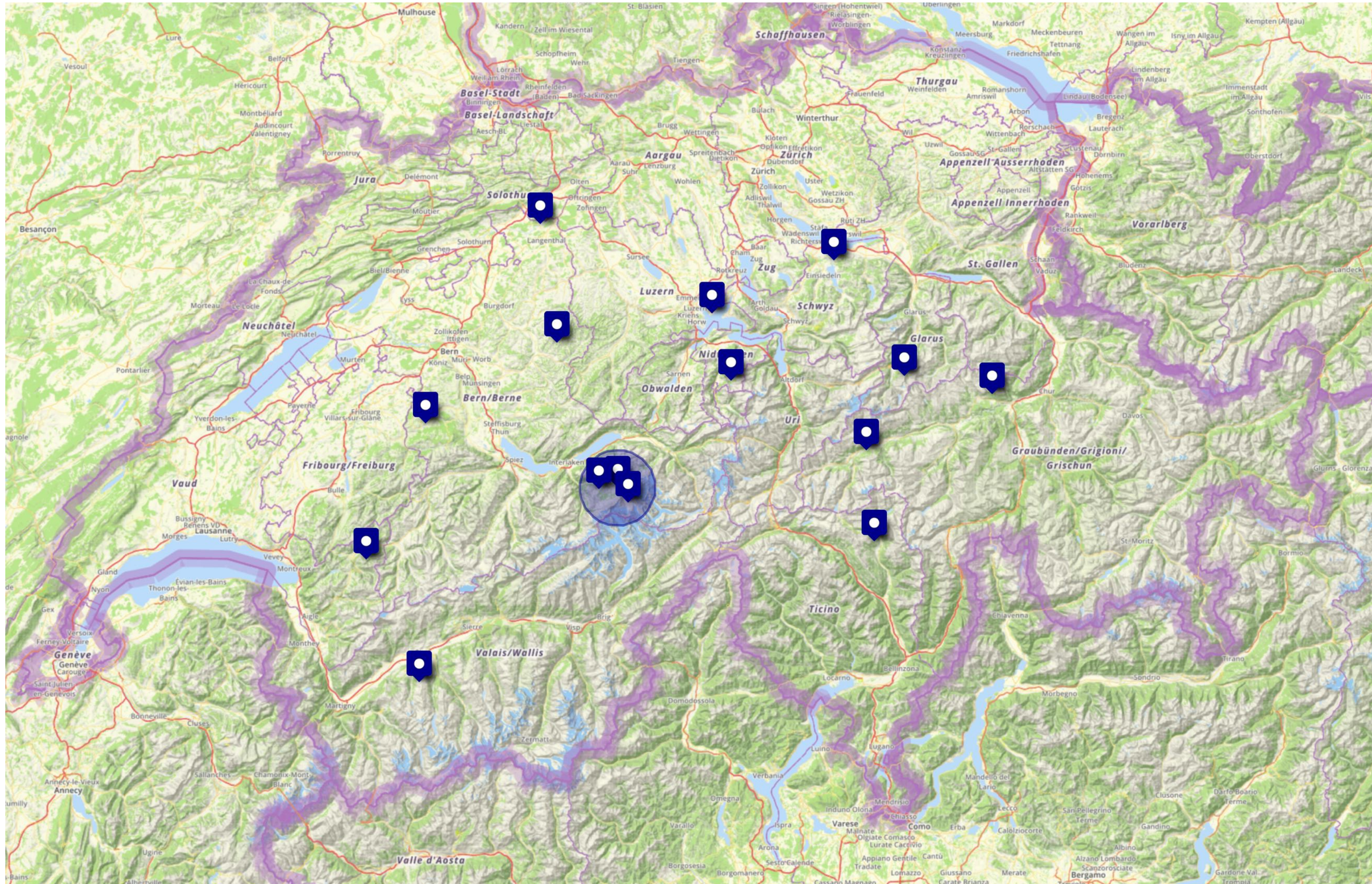
8.033339,

d.coordinates_geo_lat,

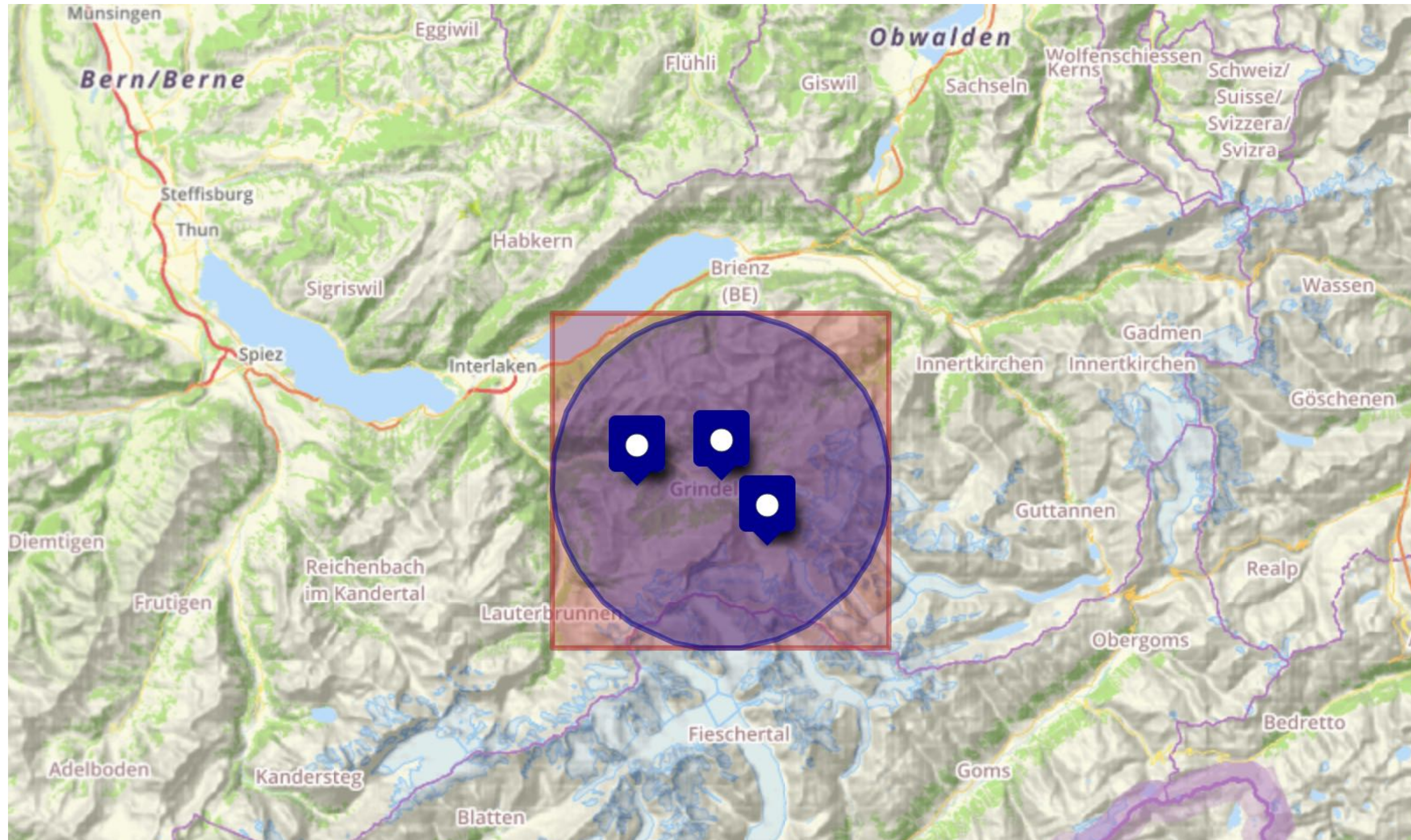
d.coordinates_geo_lng

) <= 10000

Performance-Problem



Filtern: Geo-Distanz



Filtern: Geo-Distanz

WHERE

```
d.coordinates_geo_lng BETWEEN 7.9023877390537 AND 8.1642902609463  
AND d.coordinates_geo_lat BETWEEN 46.535896966451 AND 46.715761033549  
AND loupe_geo_distance(  
    46.625829,  
    8.033339,  
    d.coordinates_geo_lat,  
    d.coordinates_geo_lng  
) <= 10000
```



Königsdisziplin

Tippfehler

Natural Language Processing

- Tokenization
- Stemming / Lemmatization
- Part-of-Speech (POS)-Tagging / Named Entity Recognition (NER)
- Beste Programmiersprache für NLP: Python
 - Tools wie SpaCy, Stanza

Tokenization (intl)

```
$it = IntlBreakIterator::createWordInstance('de_DE');  
$it->setText('Gemütliche Berghütte mit atemberaubenden Blick.');
```

```
$tokens = [];  
foreach ($it->getPartsIterator() as $part) {  
    $tokens[] = $part;  
}
```

```
print_r($tokens);
```

```
/**  
 * Array  
 * (  
 *     [0] => Gemütliche  
 *     [1] =>  
 *     [2] => Berghütte  
 *     [3] =>  
 *     [4] => mit  
 *     [5] =>  
 *     [6] => atemberaubenden  
 *     [7] =>  
 *     [8] => Blick  
 *     [9] => .  
 * )  
 */
```

Datenbank-Aufbau

Items Queries History

Search for item...

documents
info
multi_attributes
multi_attributes_documents
sqlite_sequence
terms
terms_documents

| id | term |
|----|-----------------|
| 1 | grindelwald |
| 3 | panorama |
| 5 | stube |
| 7 | stub |
| 9 | gemütliche |
| 11 | gemut |
| 13 | berghütte |
| 15 | berghutt |
| 17 | mit |
| 19 | atemberaubender |
| 21 | atemberaub |
| 23 | blick |
| 25 | auf |
| 27 | die |
| 29 | nordwand |
| 31 | des |
| 33 | eigers |

Items Queries History

Search for item...

documents
info
multi_attributes
multi_attributes_documents
sqlite_sequence
terms
terms_documents

| term | document | attribute | position |
|------|----------|-----------|----------|
| 1 | 1 | name | 1 |
| 1 | 1 | teaser | 13 |
| 1 | 9 | teaser | 16 |
| 3 | 1 | name | 2 |
| 5 | 1 | name | 3 |
| 7 | 1 | name | 3 |
| 9 | 1 | teaser | 1 |
| 11 | 1 | teaser | 1 |
| 13 | 1 | teaser | 2 |
| 15 | 1 | teaser | 2 |
| 17 | 1 | teaser | 3 |
| 17 | 5 | teaser | 3 |
| 17 | 9 | teaser | 12 |
| 19 | 1 | teaser | 4 |
| 21 | 1 | teaser | 4 |

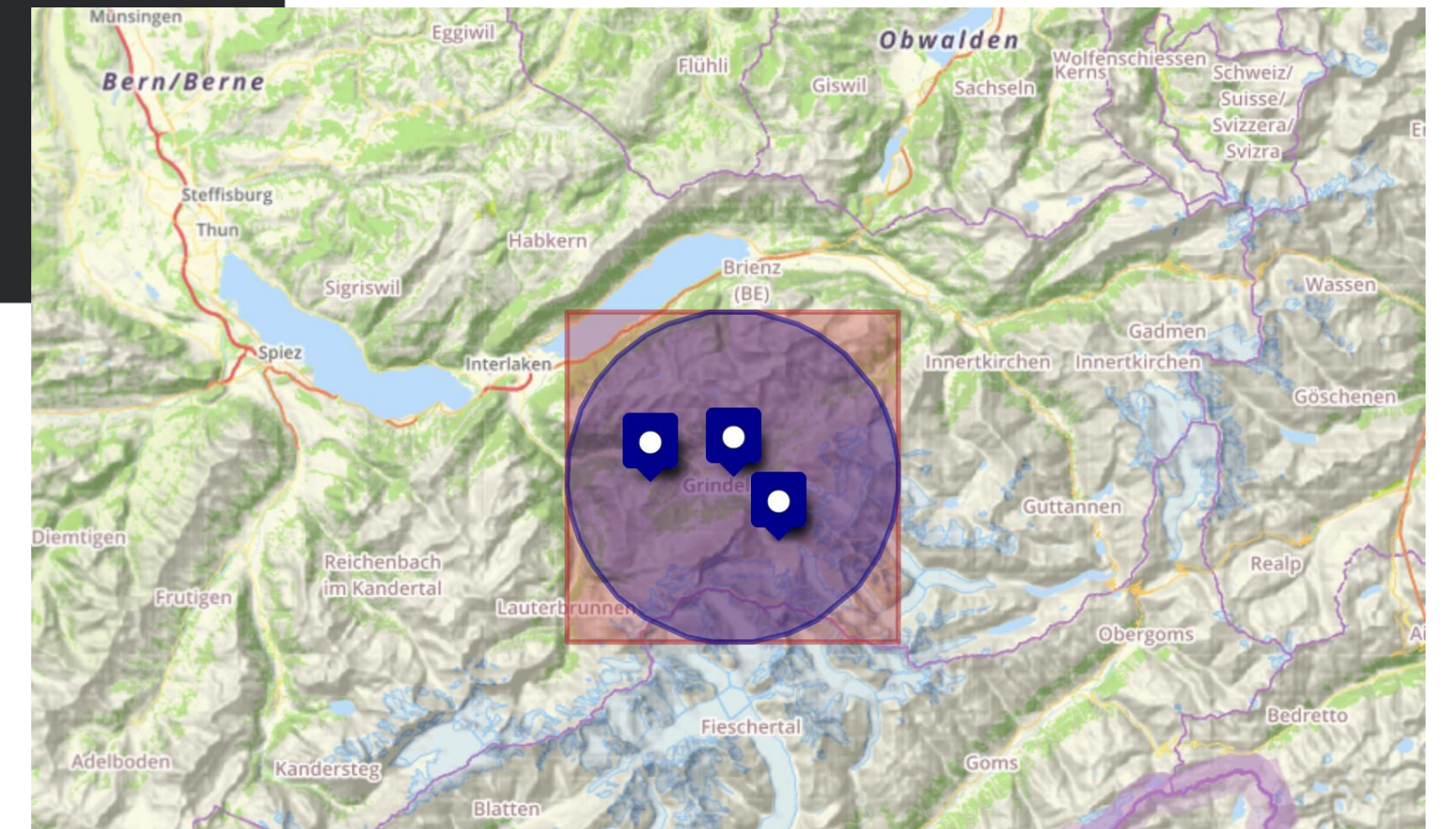
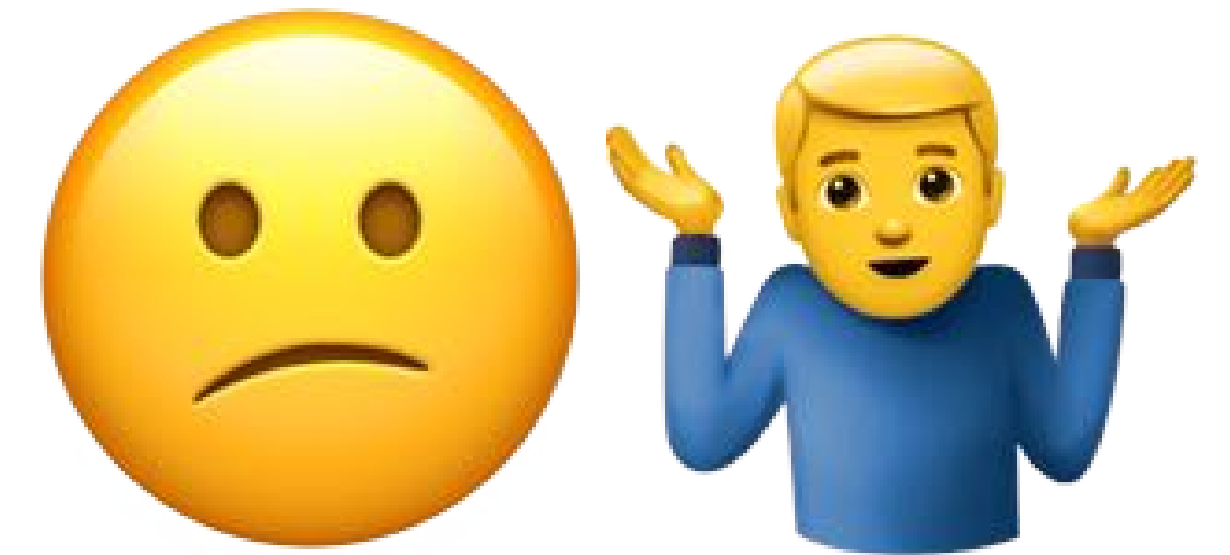
Query und Performance



```
SELECT
  td.document
FROM
  terms_documents td
  JOIN terms t ON t.id = td.term
WHERE
  loupe levenshtein('Grindlewald', t.term) <= 2
```

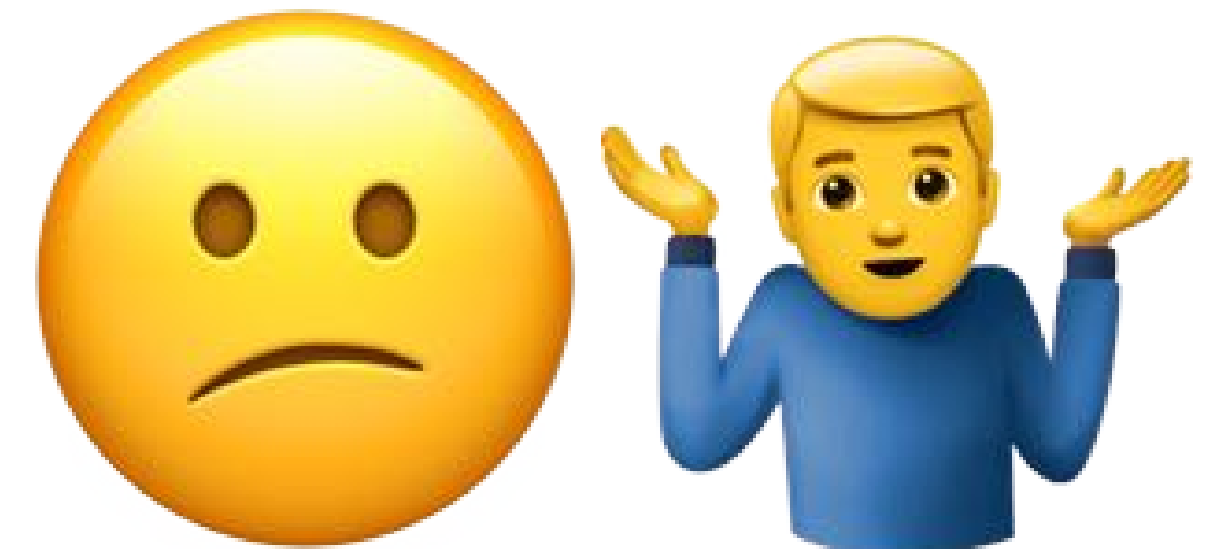


```
levenshtein('Grindlewald', 'Kandersteg'); // 9
```



Optimierungsmöglichkeiten

- Länge des Terms mitspeichern (11 bei «grindelwald») und
`length >= 9 AND length <= 13` ✓
- **Erster Buchstabe** muss zutreffen
`term LIKE 'g%' AND loupe levenshtein(...)`



State Set Index

- 2012
- <https://github.com/Toflar/state-set-index>

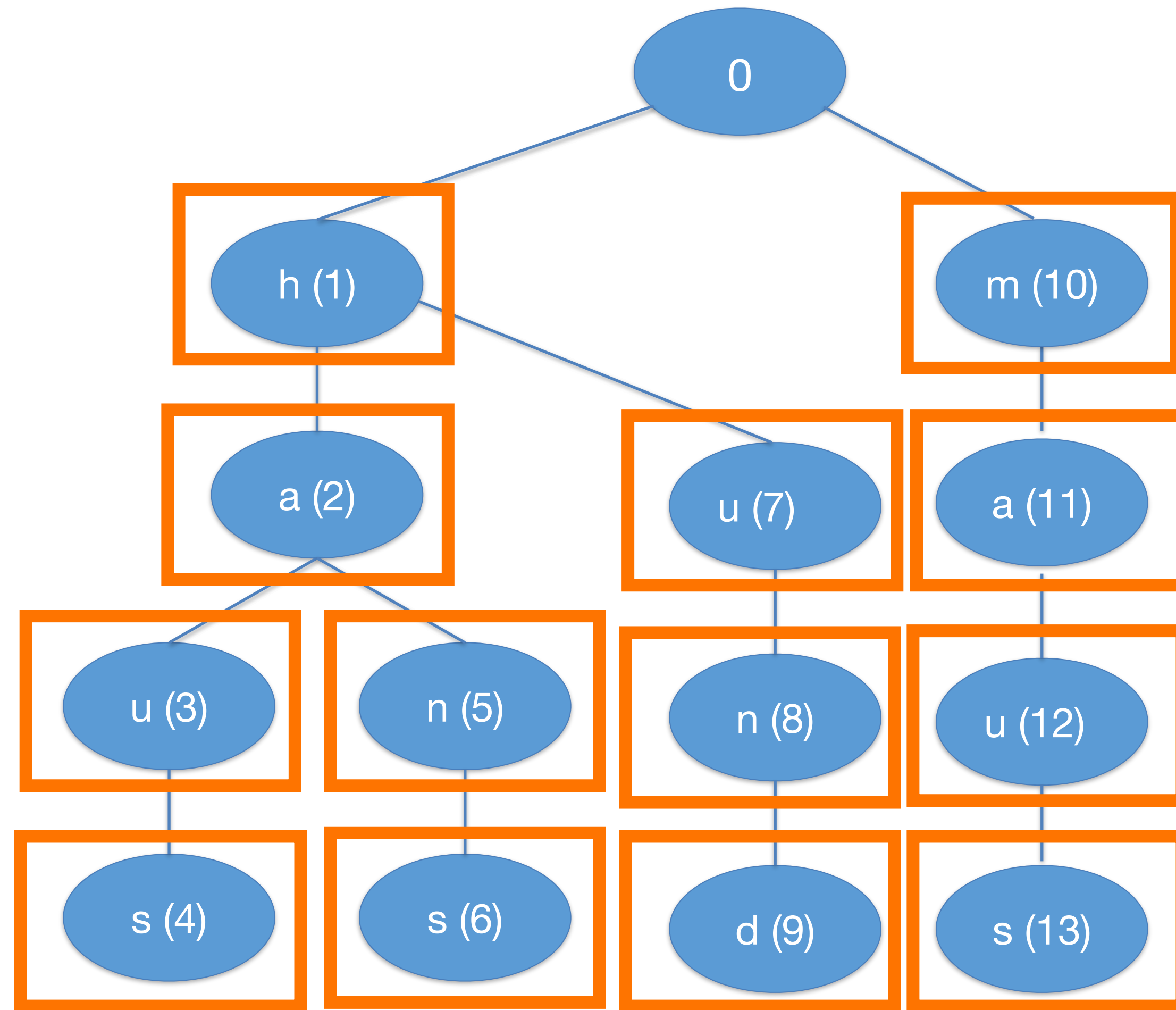
Efficient Similarity Search in Very Large String Sets

Dandy Fenz¹, Dustin Lange¹, Astrid Rheinländer², Felix Naumann¹,
and Ulf Leser²

¹ Hasso Plattner Institute, Potsdam, Germany

² Humboldt-Universität zu Berlin, Department of Computer Science, Berlin,
Germany

Automaton - Trie



- Index:

- **haus**

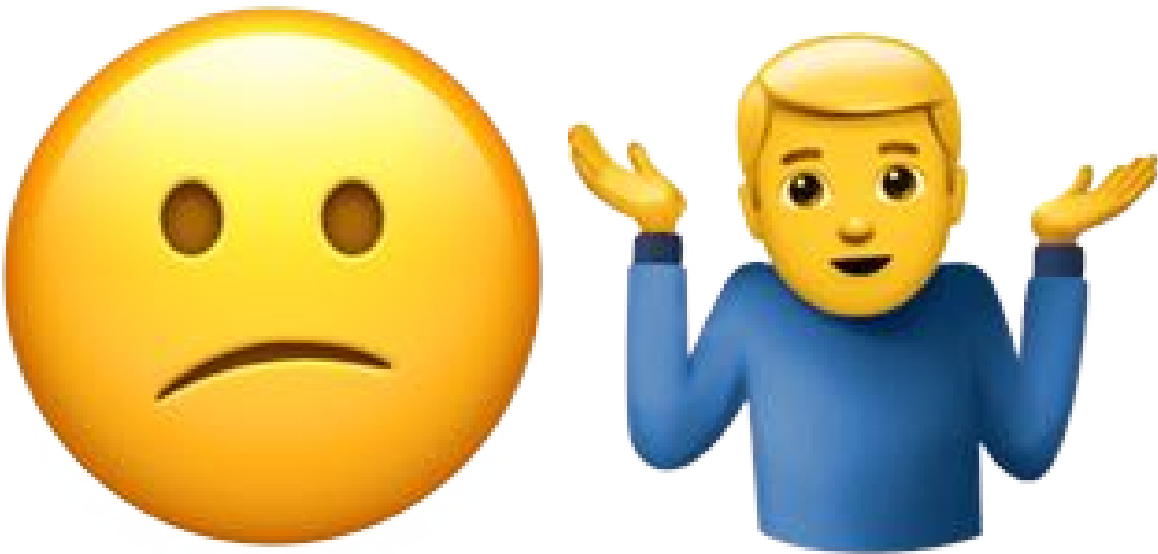
- **maus**

- **hans**

- **hund**

 «Zustand»

| Letter | State | Parent State |
|--------|-------|--------------|
| o | 0 | — |
| h | 1 | 0 |
| a | 2 | 1 |
| u | 3 | 2 |
| s | 4 | 3 |
| n | 5 | 2 |
| s | 6 | 5 |
| u | 7 | 1 |
| n | 8 | 7 |
| d | 9 | 8 |
| m | 10 | 0 |
| a | 11 | 10 |
| u | 12 | 11 |
| s | 13 | 12 |



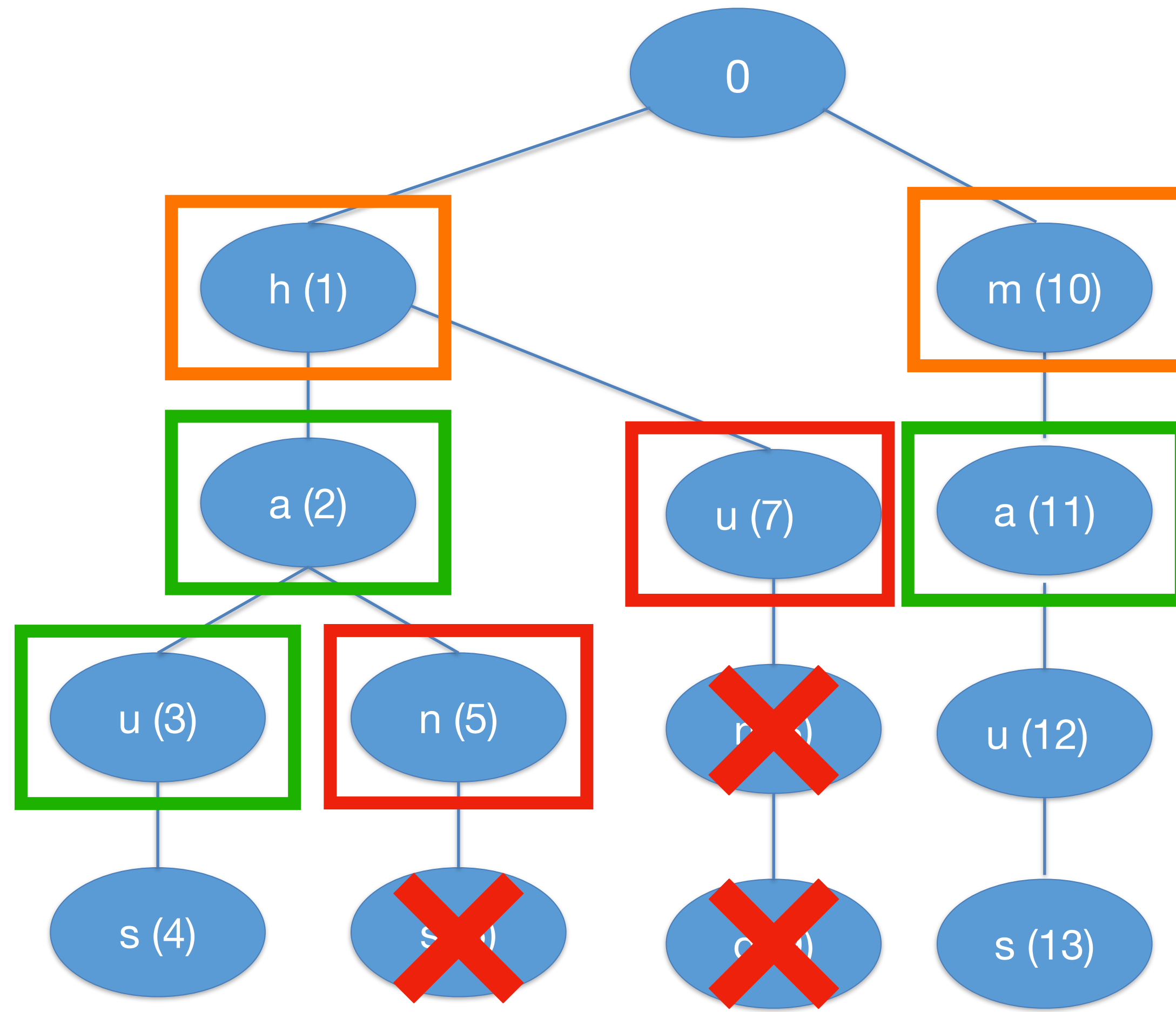
Automaton - Trie

- Wir suchen nach «**laus**» und erlauben 1 Tippfehler

WHERE parent = 0

WHERE parent IN (1, 10)

WHERE parent IN (2, 11)



l

a

u

s

SSI - Cleverness

Das Ziel ist **nicht** `levenshtein()` zu **ersetzen** sondern möglichst schnell einen **Grossteil** des Sets **auszuschliessen**.

Wir tolerieren also **False-Positives** und prüfen noch einmal final. 🧐💡

SSI - Cleverness

Wir tun einfach so, als würde unser Alphabet nur
aus 4 Buchstaben bestehen 🧐💡

Wir speichern nur ob ein Zustand erreicht wurde,
nicht wie 🧐💡

Alphabet - Unicode

| Buchstabe | Unicode Codepoint | Formel: (Codepoint % 4) + 1 | Label |
|-----------|-------------------|-----------------------------|-------|
| h | 104 | $(104 \% 4) + 1$ | 1 |
| a | 97 | $(97 \% 4) + 1$ | 2 |
| u | 117 | $(117 \% 4) + 1$ | 2 |
| s | 115 | $(115 \% 4) + 1$ | 4 |
| m | 109 | $(109 \% 4) + 1$ | 2 |
| n | 110 | $(110 \% 4) + 1$ | 3 |
| d | 100 | $(100 \% 4) + 1$ | 1 |

SSI - Kompression

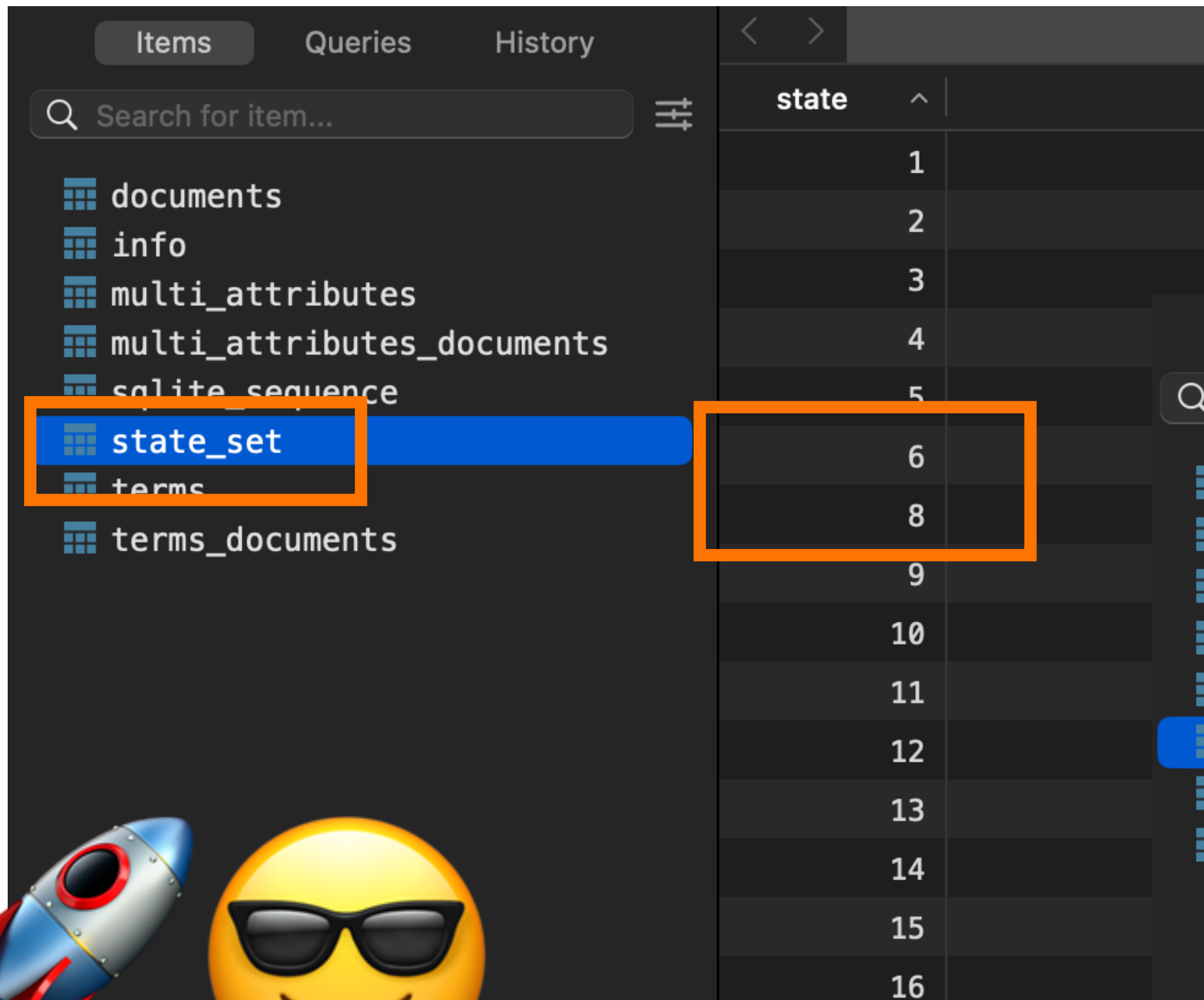
- Grosses Alphabet -> sehr **kleines Alphabet**
- **Viel weniger** mögliche **Zustände**
- **Kollisionen** sind unvermeidbar
- -> **False-Positives** die wir noch herausfiltern müssen
- Index Length: Wir schauen nur die ersten **n** Buchstaben eines Wortes an und den Rest ignorieren wir
- **Grind**ewald (7 statt 11)

| Wort | Labels |
|------|-----------|
| haus | [1,2,2,4] |
| haas | [1,2,2,4] |

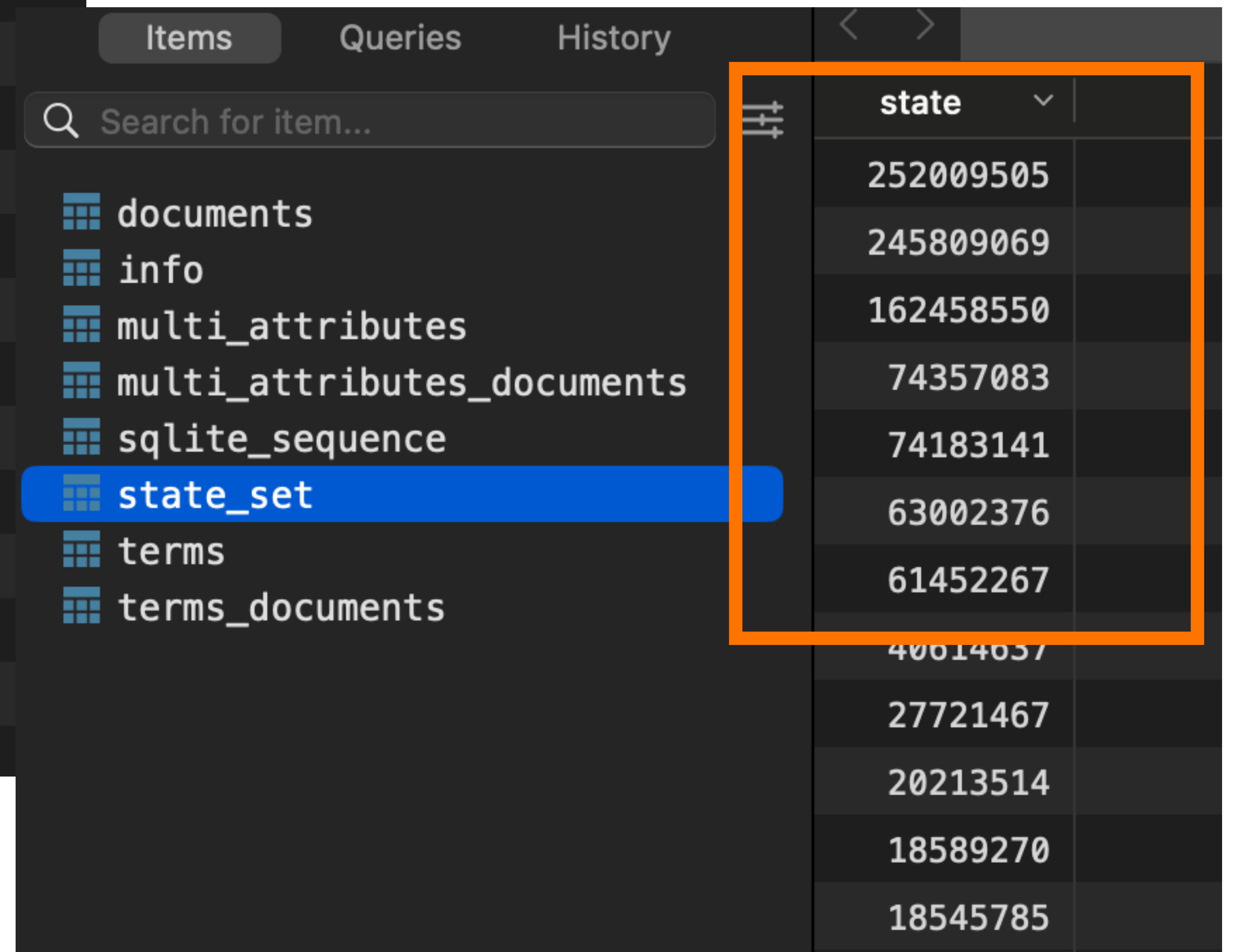
Index

$50.000 \times 8 \text{ bytes} \approx 390 \text{ KiB}$

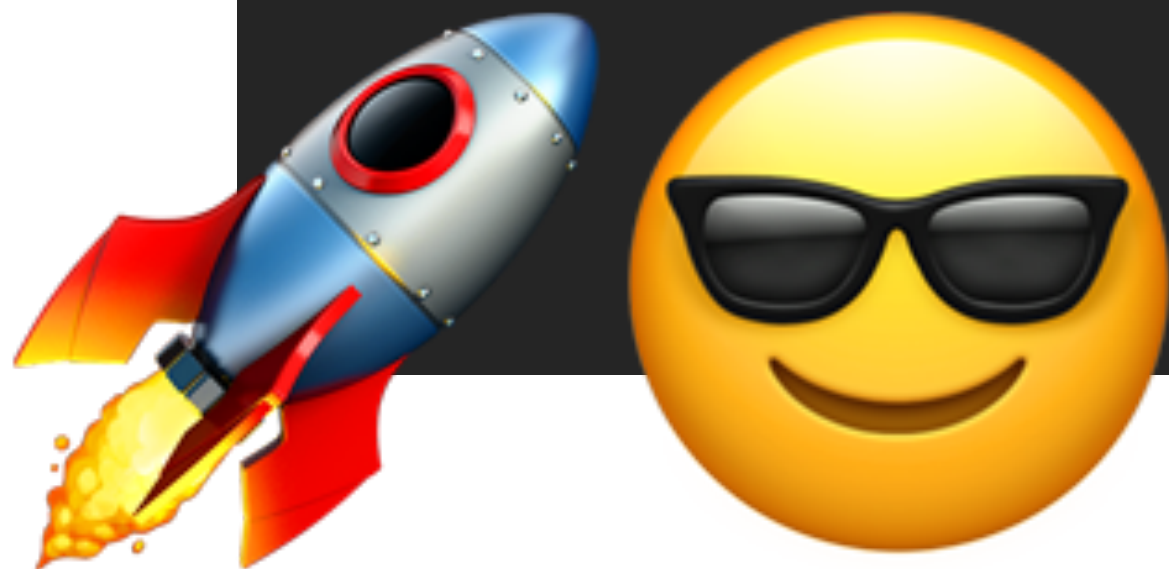
$50.000 \times 32 \text{ bytes} \approx 1.6 \text{ MiB}$



| | state |
|--|-------|
| | 1 |
| | 2 |
| | 3 |
| | 4 |
| | 5 |
| | 6 |
| | 8 |
| | 9 |
| | 10 |
| | 11 |
| | 12 |
| | 13 |
| | 14 |
| | 15 |
| | 16 |



| | state |
|--|-----------|
| | 252009505 |
| | 245809069 |
| | 162458550 |
| | 74357083 |
| | 74183141 |
| | 63002376 |
| | 61452267 |
| | 40014037 |
| | 27721467 |
| | 20213514 |
| | 18589270 |
| | 18545785 |



Index

| Items Queries History | | | SQL Query | | |
|----------------------------|--|--|-----------|--|--|
| Search for item... | | | | | |
| documents | | | | | |
| info | | | | | |
| multi_attributes | | | | | |
| multi_attributes_documents | | | | | |
| sqlite_sequence | | | | | |
| state_set | | | | | |
| terms | | | | | |
| terms_documents | | | | | |

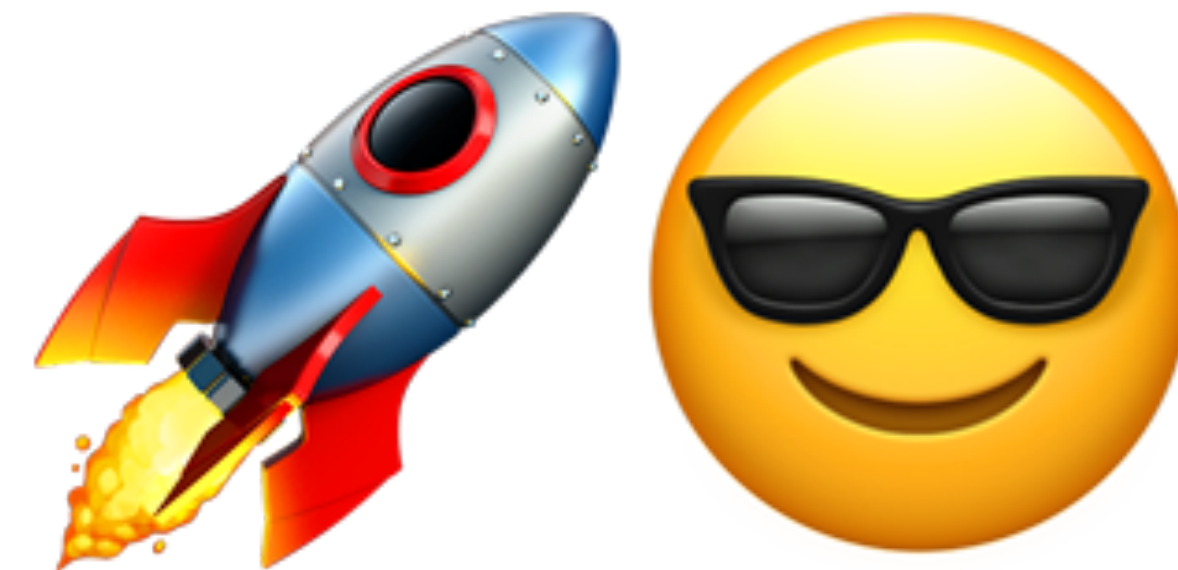
| id | term | state | length |
|----|-----------------|-----------|--------|
| 1 | grindelwald | 5167653 | 11 |
| 3 | panorama | 28906 | 8 |
| 5 | stube | 1134 | 5 |
| 7 | gemütliche | 1217990 | 10 |
| 9 | berghütte | 246102 | 9 |
| 11 | mit | 41 | 3 |
| 13 | atemberaubendem | 162458550 | 15 |
| 15 | blick | 884 | 5 |
| 17 | auf | 43 | 3 |

Levenshtein Operations

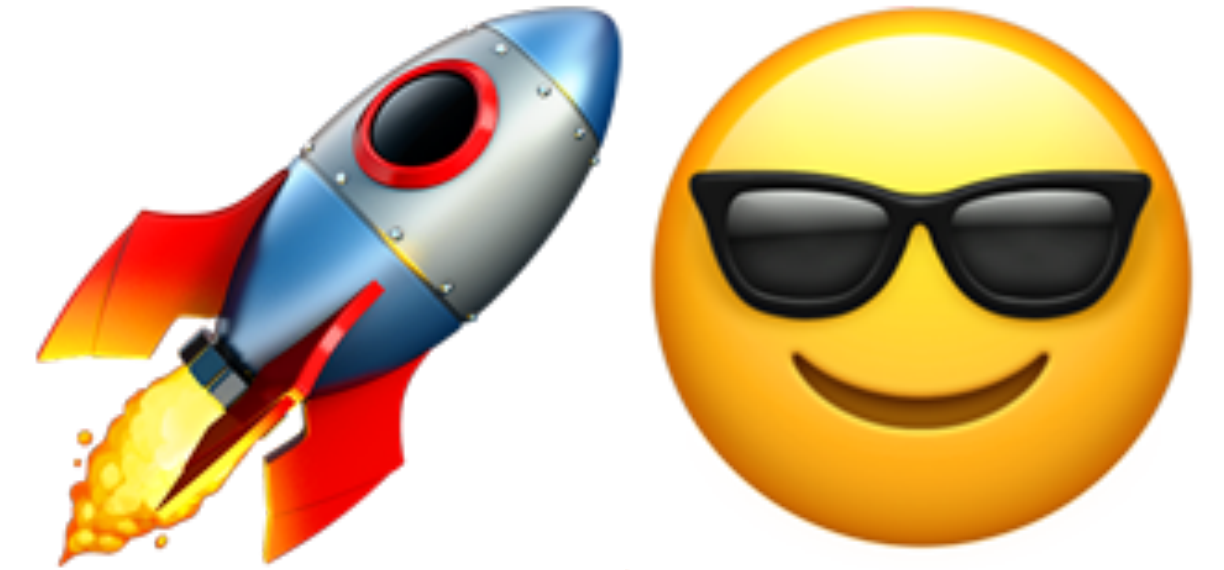
- **Match** -> Wenn der Buchstabe zutrifft, verfolge diesen Pfad
 - Mögliche weitere Pfade: **1**
- **Replace** -> Wir versuchen das Ersetzen mit allen anderen Buchstaben
 - Mögliche weitere Pfade: max. **3**
- **Insert** -> Der aktuelle Buchstabe ist zu viel im Vergleich zum passenden Wort
 - Mögliche weitere Pfade: **1** (weil «bleib wo du bist»)
- **Delete** -> Der aktuelle Buchstabe fehlt im Vergleich zum passenden Wort
 - Mögliche weitere Pfade: **4** (alle Kind-Zustände, die möglich wären)

Query

- Loop über jeden Buchstaben von «grindlewald»
- bzw. nur bspw. die **ersten 7**
- Berechnung von erreichbaren Zuständen für alle 4 Levenshtein-Operationen für alle **4 (Alphabet)** Möglichkeiten:
- { 1291913, 5167653 }



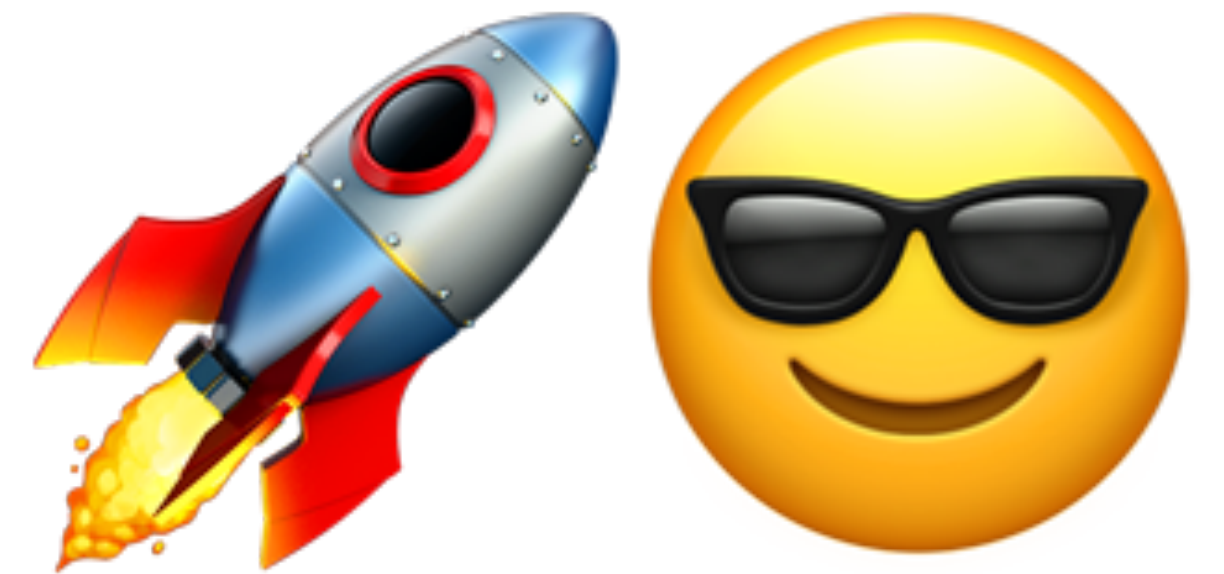
Loupe



```
SELECT
  td.document
FROM
  terms_documents td
  JOIN terms t ON t.id = td.term
WHERE
  t.length >= 9
  AND t.length <= 13
  AND t.state IN (1291913, 5167653)
  AND loupe_levenshtein('Grindwald', t.term) <= 2
```

Performance

- Test-File «movies.json» von Meilisearch (16.2 MB)
 - 32k Filme mit Titel und Teaser
 - 77k «terms» für Loupe
 - 52k Zustände für den SSI
- «loupe.db» ist ca. 210 MB gross
- Suche nach «Amakin Dkywalker» braucht 110ms und 12.50 MB RAM inklusive Relevanz-Berechnung und Facetten



Loupe kann noch mehr

- Damerau-Levenshtein («Grindlewald» vs. «Grindelwald» wäre also nur 1 Tippfehler)
- Highlighting
- Phrase-Search («"Contao Konferenz"»)
- Ausschluss-Queries mit «-»
- Stemming und Sprach-Erkennung mit N-Gramme für bessere Tokenization
- Relevanzberechnung (Anzahl zutreffende Wörter/Terms, Anzahl Tippfehler, Nähe der Matches (Position), Attribut-Gewichtung, Genauigkeit)
- Facetten (Statistiken über das aktuelle Suchresultat)





```

WITH _cte_term_matches_0 (id) AS (SELECT t.id FROM terms t WHERE (t.term = :dcValue1 OR t.state IN
(683,690,691,715,1739,2507,2699,2715,2731,2739,2747,2759,2761,2762,2763,2764,2766,2767,2771,2827,2859,2891,3019,3275,3787,4811,6859,106
99,10955,11019,11035,11047,11051,11053,11054,11056,11067,11083,11211,11467,11979,15051,19147) AND t.length >= 5 AND t.length <= 7 AND
loupe_max_levenshtein(:dcValue1, t.term, 1, true))) , _cte_term_matches_1 (id) AS (SELECT t.id FROM terms t WHERE (t.term = :dcValue2
OR t.state IN
(6795,7307,7323,7331,7371,8843,8859,8866,8867,8907,8987,8994,8995,8998,9000,9003,9010,9035,9355,9371,9379,9419,19083,19595,19611,19618,
19619,19659,21131,23707,27810,27851,28299,29131,29275,29291,29307,29327,29331,29387,29451,29467,29851,29858,29899,31371,31899,31907,352
27,35234,35275,35419,35435,35467,35468,35470,35475,35531,35611,35659,35931,35939,35947,35954,35975,35978,35979,35994,36001,36011,36019,
36034,36038,36041,36042,36043,36066,37019,37027,37323,37467,37483,37491,37539,37659,37771,38043,38050,45707,52379,52427,68747,68771,728
59,72907,76427,77451,78427,78443,78473,78475,78478,85154,85195,111195,111243,111251,111387,117131,119387,119403,119411,119443,119499,11
9563,119579,119627,126603,127435,127595,127755,136843,138379,140059,141851,141870,142027,142107,143758,143763,143817,143915,143918,1439
47,143979,144005,144078,144137,144139,144154,144174,144268,144539,144667,152201,152347,152395,275086,291475,291611,291979,313901,314251
,337547,340619,477771,536203,565851,575099,575917,576557,576619,592459,608795,1100347,1192731,1360523,4665995) AND t.length >= 8 AND
t.length <= 10 AND loupe_max_levenshtein(:dcValue2, t.term, 2, true)) OR (t.term = :dcValue3 OR t.state IN
(1832,2216,2248,2249,2252,2344,4904,6952,7464,7976,8808,8872,8984,8988,8998,9000,9004,9008,9009,9012,9016,9256,9384,9512,13096,17192,18
216,21288,35880,36001) AND t.length >= 6 AND t.length <= 8 AND loupe_max_levenshtein(:dcValue3, t.term, 1, true)) OR (t.term LIKE
:dcValue4)) , _cte_term_documents_0 (document) AS (SELECT DISTINCT td.document FROM terms_documents td WHERE td.term IN (SELECT id FROM
_cte_term_matches_0) LIMIT 1000) , _cte_term_documents_1 (document) AS (SELECT DISTINCT td.document FROM terms_documents td WHERE
td.term IN (SELECT id FROM _cte_term_matches_1) LIMIT 1000) , _cte_term_document_matches_0 (document,term,attribute,position,typos) AS
(SELECT td.document, td.term, td.attribute, td.position, loupe_levensthein(t.term, :dcValue1, true) AS typos FROM terms_documents td
INNER JOIN terms t ON t.id = td.term WHERE ((td.document IN (SELECT document FROM _cte_term_documents_0) OR td.document IN (SELECT
document FROM _cte_term_documents_1))) AND (td.term IN (SELECT id FROM _cte_term_matches_0)) ORDER BY position) ,
_cte_term_document_matches_1 (document,term,attribute,position,typos) AS (SELECT td.document, td.term, td.attribute, td.position,
loupe_levensthein(t.term, :dcValue2, true) AS typos FROM terms_documents td INNER JOIN terms t ON t.id = td.term WHERE ((td.document IN
(SELECT document FROM _cte_term_documents_0) OR td.document IN (SELECT document FROM _cte_term_documents_1))) AND (td.term IN (SELECT
id FROM _cte_term_matches_1)) ORDER BY position) , _cte_matches (document_id) AS (SELECT DISTINCT document_id FROM (SELECT d.id AS
document_id FROM documents d WHERE ((d.id IN (SELECT DISTINCT document FROM _cte_term_document_matches_0)) OR (d.id IN (SELECT DISTINCT
document FROM _cte_term_document_matches_1))))) , _facet_count_genres (facet_group,facet_value) AS (SELECT ma.string_value,
COUNT(DISTINCT mad.document) FROM multi_attributes_documents mad INNER JOIN multi_attributes ma ON ma.attribute=:dcValue5 AND ma.id =
mad.attribute INNER JOIN _cte_matches _cte_matches ON _cte_matches.document_id = mad.document WHERE (ma.string_value!= :dcValue6) AND
(ma.string_value!= :dcValue7) GROUP BY ma.string_value LIMIT 100) , relevances_per_document_term_0 (document_id,relevance_per_term) AS
(SELECT _cte_matches.document_id AS document, COALESCE(group_concat(DISTINCT dm.position || ':' || dm.attribute || ':' || dm.typos),
'0' ) AS relevance FROM _cte_matches LEFT JOIN _cte_term_document_matches_0 dm ON dm.document = _cte_matches.document_id GROUP BY
_cte_matches.document_id) , relevances_per_document_term_1 (document_id,relevance_per_term) AS (SELECT _cte_matches.document_id AS
document, COALESCE(group_concat(DISTINCT dm.position || ':' || dm.attribute || ':' || dm.typos), '0' ) AS relevance FROM _cte_matches
LEFT JOIN _cte_term_document_matches_1 dm ON dm.document = _cte_matches.document_id GROUP BY _cte_matches.document_id) ,
relevances_per_document (document_id,relevance_per_term) AS (SELECT _cte_matches.document_id AS document,
relevances_per_document_term_0.relevance_per_term || ';' || relevances_per_document_term_1.relevance_per_term AS relevance_per_term
FROM _cte_matches LEFT JOIN relevances_per_document_term_0 relevances_per_document_term_0 ON relevances_per_document_term_0.document_id
= _cte_matches.document_id LEFT JOIN relevances_per_document_term_1 relevances_per_document_term_1 ON
relevances_per_document_term_1.document_id = _cte_matches.document_id) SELECT d.document, (SELECT GROUP_CONCAT(facet_group || ':' ||
facet_value) FROM _facet_count_genres) AS _facet_count_genres, MIN(1000, COUNT() OVER()) AS totalHits,
loupe_relevance('title:overview', 'words:typo:proximity:attribute:exactness', relevances_per_document.relevance_per_term) AS _relevance
FROM documents d INNER JOIN _cte_matches _cte_matches ON d.id = _cte_matches.document_id INNER JOIN relevances_per_document
relevances_per_document ON relevances_per_document.document_id = _cte_matches.document_id ORDER BY _relevance DESC LIMIT 20

```

Links

- <https://github.com/loupe-php/loupe>
- <https://github.com/PHP-CMSIG/search>
- <https://github.com/PHP-CMSIG/seal-loupe-adapter>
- <https://github.com/sponsors/Toflar>



Kontakt

- Fediverse: phpc.social/@toflar
- X: [@toflar](https://twitter.com/toflar)
- GitHub: [toflar](https://github.com/toflar)
- E-Mail: yanick@terminal42.ch
- Slack

