Unified Index

As of Tiki6, global search can be made through either a custom search indexing implementation or through MySQL's full text indexing. Both search features have different problems and come to hit limitations when additional filtering is required. For example, searching by language or by category requires several joins that slow down the execution significantly. Some features become unusable beyond a certain volume or require excessive resources on the database.

The problem of searching spreads further. Each feature has its own listing page and wiki plugins. Those plugins do not benefit from full text indexing and all of those come with different arguments. The differences between features cause severe inconsistencies for end users, leading to multiple feature requests and bug reports. One example is category filters. While categories are a global feature, the types of filtering available for each feature varies and the implementations are simply duplications of ad-hoc queries built.

The unified index proposes to defer listing and filtering to a specialized engine rather than using the database directly for all object listings for the end users. The relational model simply cannot perform some of the requested queries efficiently. Yet, SQL will allow the filtering to happen and will work in test environments. Moving to an external engine will force development to consider the efficiency. Better code organization will also lead global features to be available evenly. Internally, Tiki could still use direct function calls to the database when appropriate, but the advanced features should be removed from the libraries to clean up the code.

Target benefits:

- Improved search results
- Improved consistency across features
- Improved efficiency for complex queries
- Code reuse across features
- Simplify the codebase

As an initial implementation, Zend_Search_Lucene is proposed as the indexing engine. The library is mature and already bundled with Tiki. Because it is written in PHP, there are no external dependencies and can work in multiple environments. The solution is expected to be sufficient for most Tiki installations. A quick search indicated that limitations were hit around indexes of around 200M when search quality was not sacrificed. Larger sites like dev.tiki.org or doc.tiki.org should fit well below this.

However, the engine has limitations that come with the PHP implementation, like memory consumption for large installations. For this reason, the unified index will work through an abstraction layer that would allow integration with different indexers for larger installations (Sphinx, Solr, ...). Those engines will have different attributes when it comes to search performance, indexing performance, scalability.

Indexing

There will be a single index for all object types. Obtaining a list for a single type will be about adding a filter. The indexing mechanism needs to reflect the structure of Tiki. For each feature to be indexed, an adapter will be required to pull the basic information, like page titles, description, content, modification date and so on.

Essentially all the indexable information coming from the database table. All fields for which sorting or searching is desired will need to be added to the index. Because similar information often has different names in Tiki, it will be important to impose a naming convention to make sure the index is usable.

Before being indexed, the basic information will be augmented with the global features, like categories, comments, attachments and others. This will guarantee consistency across features. Once implementation is completed, the category filters would apply the same way between wiki pages and blog posts. Future feature additions in the search index will also apply to all features without having to go through each feature individually.

Additional transformations can also be performed globally, like splitting camel case words so that searching for a single word would return results. Some global behaviors may also be implemented here, like wiki-parsing the wiki fields before indexing. The objective is to regroup site-wide policies in a single point to relieve the different features from those decisions. If it is determined that a special group need to be used to perform the wiki parsing for indexing, there would be a single location to update.

Information would then be provided to the indexing abstraction so that the index is populated in an optimal way. To allow future expansions, it will be important for the index builders not to assume the presence of some specific engine feature.

Indexing will be done either on update or batch (cron or on demand), depending on configuration or capabilities of the indexer. Larger sites may not be able to handle the constant index optimization required by incremental updates and require to do it offline from a database slave or something similar. Incremental updates will likely hook into the current searchlib.

Stemming will be relegated to the indexing engine. Lucene has a pluggable interface for those and the codebase already includes a stemmer for English (used in the configuration search). It may be required to build multiple search indexes to use different stemmers. Each index would contain the exact same content from all languages regardless.

Plugin

The unified index will render multiple plugins obsolete. For backwards compatibility, some will need to be modified to use the alternative library. To better support the transition, they may be left untouched.

One of the main issues at the moment with the plugins is that they have a very large amount of parameters. Many of those parameters were added for very specific use cases. Sometimes, parameters that seem equivalent in two plugins will behave in very different way. Not to mention the inconsistencies in separators (colon, semi-colon, comma, plus, space, ...). The main issue is that the plugin parameter format does not allow for enough expressiveness.

Modifications to the parser could allow for a more readable format. Progress was already made on the plugin parser to invert the execution order. However, for historical reasons, it was never integrated. Now that we are past an LTS, it may be a good time to complete the parser and replace it. Doing so would allow to write the listing plugin in the following manner.

As implemented - NOT A SPECIFICATION

```
{LIST()}
```

```
  {filter type="wiki page"}
  {filter categories="(1 and 2 and 3) or (4 and 5)"}
  {filter deepcategories="25 or 26"}
  {filter language="en or unknown"}
  {filter content="hello world"}

  {sort mode=modification_date_ndesc}

  {OUTPUT()}* {display name=modification_date format=date}: {display name=object_id}
{OUTPUT}
{LIST}
```

In the above, LIST is the plugin and it reads the content to identify the parameters of the query. Some arguments could be added on list for very common operations, but anything beyond trivial should be made through the configuration in the body. The body could also provide ways to specify a template for how to display the results and such.

To keep the implementation clean on the long run, the list plugin should limit the amount of primitives it support and use macro expansion for other tasks (essentially, using the plugin parser with a different plugin directory, and those plugins would output primitives for evaluation by the list). The syntax needs to form an understandable language that can be composed to perform advanced reports. Adding new features results in adding new vocabulary that can be used beyond the very specific use case for which it was created.

Internally, the plugin would generate a query object that it would send to the indexer abstraction to obtain the results and then format them for output. The same query object could be built from PHP to generate the tiki-listpages.php page or the search results from a search query. The default result formatter could also be used to render the list in those pages.

Multiple sorting modes would be available from the Tiki data, including advanced ratings. However, the ranking based on result relevance will be relegated to the engine. The engine's syntax often contains tricks that can be used to adjust the relevance. Those may be used when building the query for the engine based on the user's input.

Boolean Query Syntax

Traditionally, Tiki exposed the engine's syntax directly with all the features it contain. There is a significant amount of documentation explaining all the quirks of the MySQL full text search syntax which no normal user would ever use. Because multiple engines will be supported in the future, Tiki should commit to a minimal syntax that is frequently used. The syntax should be usable across engines, even if transformation is required on Tiki's end.

The proposed syntax is the following:

Query	**Description**
some search terms	OR A wish to have a AND matching
some **OR** search **OR** terms	OR matching
"some search term"	exact match
some **+** search **+** term	AND matching

some **AND** search **AND** term	AND matching
some search **AND** terms	(some OR (search AND terms))
some **OR** search **AND** terms	(some OR search) AND terms
some **OR (**search **AND** terms**)**	some OR (search AND terms)
some **NOT** search	some OR NOT(search)
some **AND NOT** search	some AND NOT(search)

Token priority: Quotes > Parenthesis > NOT > OR > AND > + > space

Note on exact match
The exact match is done on stemmed token and do not considered stopwords or short words.
A query "PHP framework" with the double quotes will find "PHP the framework" or "php frameworks", but not "framework php" or "php the best framework". A query "PHP the frameworks" will find "php framework".

Result Ordering

Tiki has a unified way of indicating the search order. The syntax can be found in various URL schemes and plugins. In order to ease the transition, the syntax must be preserved. The syntax used to be the database field name followed by *_asc* or *_desc*. The syntax is being converted directly to an SQL order by statement, which may cause SQL errors when an invalid field is provided. Moreover, the sort ordering heavily depended on MySQL. Because Tiki often uses text field to store generic values, when numeric sorting is required, the user had no control over the ordering.

Proposed conversions (fields are indexed fields):

Sort Mode	**Field**	**Order**	**Sort Type**
title	title	asc	text
title_asc	title	asc	text
title_desc	title	desc	text
title_nasc	title	asc	numeric
title_ndesc	title	desc	numeric

Permission Filtering

A long standing issue was the pagination of results when filtering permissions. The database structure in which the permissions were stored and some logic only stored in the code did not allow to perform the filtering within the query. As a result, the pagination could not be made evenly without going through a lot of trouble.

To resolve this issue, the unified index indexes the groups allowed to view an object through a global source. For each object added to the index:

1. Verifies if the content source provided the view permission name
2. Obtain the permission set for the object
3. Introspect to find the groups affecting the permissions
4. Check for each group individually to see if the view permission is available

See PermissionSource::getData for more details.

During the query phase, the list of groups of the active user is used to filter the allowed groups field.

Objects can be filtered efficiently because they are indexed while the indexing time increases. Because the permissions are no longer validated just in time, permission changes may lead to WYSIWYCA rules to be broken temporarily. Objects may be listed extra and some may be missing. However, a direct access to the object will still see the up to date permissions to be validated.

Date filtering

(request by ricks99)
Need to be able to filter out expired content:

1. Calendar items that have passed
2. Articles that have expired (even if the **show after expired** type option = **y**).

Formatting results

When it comes to formatting search results, Tiki has a standard table format that is a suitable default for internal applications that benefit from a unified look and feel. However, some cases require different layouts, especially when custom designs are used. In some cases, there are simply too many values to display in tabular format. Pretty trackers are a solution to this. Unfortunately, they are part of those features that are only available to a single feature.

The unified index should allow for multiple result rendering methods, from the default table to wiki-based templates to full-blown Smarty templates for more complex cases. Essentially, all of those share a common pattern:

1. Obtain the search results
2. Gather additional information required by display not provided in results
3. Perform basic formatting on values
4. Lay out the information

Ideally, the format specification for the default model should allow for wiki formatting within the cells. For example, a column could contain multiple actual values, allowing for some customization without having to go all the way with smarty templates.

Components

The project requires a significant amount of work. A lot can be made incrementally by adding features where they belong and provide the benefit to multiple features. However, adapters still need to be written for every feature in Tiki that need to be indexed.

The primary components are:

- Indexer
  - Pluggable interface for content sources, extracting basic information from Tiki (one per content feature)
  - Pluggable interface for global sources, extracting additional, site-wide, information about a single object (one per global feature)
  - Bridge to Lucene
- Search Query
  - Internal structure definition
  - Translator to Lucene query
  - Result caching
- Formatter
  - Fetching additional information (related to content adapters and global adapters)
  - Format as generic table
  - Value formatters (dates, numbers, currency, etc)
- Plugin parser
- List plugin
  - Build query
  - Build formatter
  - Macro expansion
- Search UI
  - Multiple filters
  - As a re-usable component

As an initial step, a walking skeleton can be built to go from indexing of wiki pages to listing with a basic plugin. Then global features can be added. Additional adapters for features, expansion of the formatter and search UI can all happen in parallel once the core is in place.

Done this far:

- Indexer, aggregating information from content sources and global sources
- Query interface and transformation for Lucene
  - Query parser (described above, but currently using OR matching by default)
  - Basic type filtering
  - Full text searching, with boolean capabilities
  - Language filtering, with boolean capabilities
  - Category filtering, deep category filtering, with boolean capabilities
  - Permission filtering based on the user's groups
- Result sorting
- Formatting
  - Wiki (basic)
  - Smarty
- Result pagination
- Content sources
  - Wiki
  - Forum Post
  - Files in file galleries

    - Blog posts
    - Articles
    - Tracker items
- Global sources
    - Permission
    - Category
    - Freetags
    - Advanced Ratings
- Stemming
    - English (used as default - the Porter stemmer algorithm) + stopwords defined in lib/core/StandardAnalyzer/Analyzer/Standard/English.php
    - Some special characters as - , ?, ... are handled in lib/core/Search/Index/Lucene/php:buildTerm

How-To

Add a content source

Content sources are used to index objects of a given type (wiki pages, forum post, tracker items, ...)

1. Create a new class for the content type implementing Search_ContentSource_Interface
    - List all objects in the system for global indexing
    - Provide the information for a requested object using the provided TypeFactory to describe the type of content
    - A list of the provided fields for general introspection
2. Register the content source in unifiedsearchlib if the feature is enabled
3. Update the field matrix in the documentation

Standardized field names are provided to help cross-feature searches. As a bare minimum, the following fields must be provided:

- title
- language (with the value *unknown* if unspecified)
- modification_date

Additionally, the permissions must be specified in some way as the permission verification is done by the unified index. For typical objects, the *view_permission* field may be used to indicate the permission to check. Some object types will defer permission verification to the parent object (forum posts use permissions from the forum for example). In this case, the following fields can be provided:

- parent_object_id
- parent_object_type
- parent_view_permission

Possible User Scenarios

Multilingual

Use Case 1

- Page **Foo** is EN; page **Bar** is ES translation of **Foo**
- EN User searches for a term
- If **Foo** is returned in the results, need a way to let user know that ES translation of the page exists
    - Inform user of "up-to-dateness" of pages
- If **Foo** is *not* returned in the results, but it exists in a translated page, return the EN page with a notation that the term exists in the ES version

Weighting

When calculating search results, need to account for:

- Free tags used in the page
- Free tags used in translations of the page
- Keywords assigned to the page
- Prior search/click popularity
- Content ratings (e.g., article ratings, wiki page ratings, comment ratings, forum ratings, etc.). Items with higher ratings should be returned as better matches

Integrated Tikis

For multi-tiki sites (or TRIM, maybe too?), users should be able to specify/filter searches across multiple sites.

Modules

I (ricks99) would love to see some new Tiki modules that take advantage of the improved search:

- Top search terms
- Top results
    - Based on returned values
    - Based on click-through values

Radar

In the scenario where flagged edits became a Tiki features, it would be nice that the unified index could index last version of pages and last approved version for your group. WYSIWYCA would be respected in search results.

Bugs

- ~~r31459 tiki-searchindex.php (without arguments) shows all results instead of none~~
- r31459 "Parse the results" is activated but I see wiki syntax
  - Preference not supported at this time. Not a bug.
- Items from some trackers appear with "No title specified", although list tracker items manages to find a title. Happens on my local test install and on at least one more site. Chealer r31457
  - Need more details. Provide steps to reproduce or a profile.
  - Probably related: article "Foo" is listed as "Foo in No title specified". Chealer 20110916
- Object titles are not escaped (seen with Wiki pages). Chealer r31457
  - Code does it. lib/smarty_tiki/function.object_link.php:60
    - That doesn't seem to be called. Might also be passed misencoded data. Reproduced with ampersands on r37327. Chealer
- ~~Tag filter does not escape its value properly (heavy jQuery seems involved). Chealer r31457~~
  - Again, everything seems to be in order. Provide more details.
  - This appears to be a general freetags issue caused by quotes. Chealer 20110916
- Many objects show twice in results. Chealer r31457
  - Found a case where the documents were not being invalidated when containing special characters. Retry with r31686
- Fatal error
  ~~updating page with an exclamation mark in its name (find() called with wrong query).~~
  - Could not reproduce the exception, although it may now be resolved.
  - Couldn't reproduce on r37237. Chealer9

Feature requests

- Result snippet doesn't necessarily contain search term

Wishes

Open

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
●	★★★ ★★★ ★★★ ★★★ ★★★ ★★★ ★★★ ★★★ ★★★ ★★★ (1) ?	Monitoring pre-dogfood servers	Marc Laporte	10 high	9	90	• Feature request • Community projects • Indexing	Marc Laporte	2013-08-21	2021-07-20	5 marclaporte-20 Jul 21

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
●	(1)	Search index could not be updated. The site is misconfigured. Contact an administrator.	Marc Laporte	9	1 difficult	9	• Dogfood on a *.tiki.org site • Conflict of two features (each works well independently) • Indexing		2012-01-19	2015-05-19	2 xavi-29 May 18
●	(0)	Not possible to delete a tracker on next.tw.o	Marc Laporte	10 high	8	80	• Error • Indexing	nkoth	2013-08-15	2014-07-30	6 koth-04 Aug 14
●	(0)	Unknown column 'defaultOrderKey' in 'on clause' on http://next.tiki.org/tracker16	Marc Laporte	7	8	56	• Error • Indexing	Nelson Ko	2013-07-29	2013-11-04	11 marclaporte-24 Dec 14
●	(0)	"Error: Malformed search query" without anything else causes support issues. Add an actionable error message.	Marc Laporte	7	8	56	• Error • Usability • Indexing		2018-10-21	2018-10-21	1 marclaporte-13 Nov 18
●	(0)	MariaDB (MySQL) Unified Index doesn't permit same reports as Elasticsearch (No results for query)	Marc Laporte	7	7	49	• Indexing		2018-10-12	2018-10-12	3 luci-24 Jan 19
●	(0)	Fix Dev.T.O indexing issues after daily ugrades	Marc Laporte	9	5	45	• Indexing	Nelson Ko	2013-11-26	2013-11-27	2 koth-28 Nov 13

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
●	(3)	Trackers: Need to be able sort as numerical instead of text	Marc Laporte	8	5	40	• Error • Feature request • Dogfood on a *.tiki.org site • Conflict of two features (each works well independently) • Developer Training • Indexing • Easy for Newbie Dev		2007-12-07	2022-01-01	11 hman-01 Jan 22
●	(0)	Indexing failed while processing "Update" (type wiki page) with the error "RemoteTransportException[	Marc Laporte	8	5	40	• Error • Indexing	Nelson Ko	2013-11-30	2013-11-30	0
●	(0)	SearchPhaseExecutionException on tiki-searchindex.php with sort_mode=modification_date_ndesc	Marc Laporte	5	8	40	• Indexing		2014-08-10	2014-08-10	0
●	(0)	dev.tiki.org: indexing failed while processing "Revert a commit" (type wiki page) with the error	Marc Laporte			25	• Indexing		2014-06-26	2014-06-26	0
●	(0)	Last date of re-index can be incorrect if I just changed from Elasticsearch to MySQL Unified Index	Marc Laporte	2	5	10	• Conflict of two features (each works well independently) • Indexing		2018-10-12	2018-10-12	0

Pending

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
◑	(0)	Make Unified Index optional	Marc Laporte	10 high	8	80	• Feature request • Conflict of two features (each works well independently) • Indexing	Nelson Ko	2013-06-09	2013-11-04	3 marclaporte-22 Nov 13

Closed

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
✖	(0)	Search index error	vmchieu	9	1 difficult	9	• Error • Indexing		2011-10-30	2013-11-04	1 marclaporte-24 Nov 11
✖	(1)	Rebuilding index stats: add Execution time, Memory usage and number of Queries	Marc Laporte	9	9	81	• Feature request • Indexing	Nelson Ko	2013-10-25	2016-10-12	1 marclaporte-12 Oct 16
✖	(0)	Make Unified Index with MySQL work well and be the default in Tiki12 (if Unified Index is selected)	Marc Laporte	9	8	72	• Indexing	Nelson Ko	2013-09-08	2013-11-17	6 pascalstjean-17 Nov 13
✖	(0)	"No index available": offer a link to rebuild index	Marc Laporte	8	9	72	• Indexing	manivannans	2013-11-20	2013-12-08	1 marclaporte-21 Nov 13

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
	(0)	Login module does not remind of user name any more	Jean-Marc Libs	8	8	64	• Indexing	Jean-Marc Libs	2016-04-06	2016-04-06	0
	(0)	Error on tracker first creation - error text is confusing and wrong: Field tracker_id does not exist in the current index. If this is a tracker field, the proper syntax is tracker_field_tracker_id.	Bernard Sfez / Tiki Specialist	8	8	64	• Usability • Regression • Consistency • Indexing	Victor Emanouilov	2018-10-03	2018-10-12	1 marclaporte-04 Oct 18
	(0)	PHP Fatal error with ANY profile since 15.x: Search_MySql_LimitReachedException Incorrect datetime value: '0000-00-00 00:00:00' for column 'creation_date'	Xavier de Pedro	8	6	48	• Error • Community projects • Dogfood on a *.tiki.org site • Regression • Indexing	Jonny Bradley	2016-05-18	2016-05-26	3 xavi-24 May 16
	(1)	Unified index incremental update fails on 12.x svn (untested in 14.x)	Xavier de Pedro	9	3	27	• Error • Regression • Conflict of two features (each works well independently) • Release Blocker • Indexing		2015-05-19	2016-06-02	0
	(0)	Search issue when using CustomSearch with MySQL Full Text Search as the Index	Pascal St-Jean	5	5	25	• Error • Indexing	manivannans	2013-08-12	2013-11-12	1 xavi-12 Aug 13
	(0)	Unified Index: Add spreadsheet	Marc Laporte	5	5	25	• Feature request • Indexing		2013-11-04	2013-11-10	0

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	Created	LastModif	Comments
	(0)	need way to remove temp/preference-index-en*	Marc Laporte	5	4	20	• Conflict of two features (each works well independently) • Indexing	manivannans	2012-03-28	2013-11-12	2 jonnybradley-05 Nov 13

Tools

- to debug the lucene database, you can usee luke
    - download the jar at http://code.google.com/p/luke/ lukeall
    - for my lunix I use the command java -jar ~/Software/lukeall-1.0.1.jar
    - open the lucene directory default temp/unified-index
    - Example: to search on tiki in wiki page type in the search tab +(contents:calabrio)+(object_type:"wiki page")

Cron job and command line

You can run this php file in a cron job or on a line command to have the indexes built

```
<?php
include_once('tiki-setup.php');
require_once 'lib/search/searchlib-unified.php';
$unifiedsearchlib->rebuild();
```

In Tiki 8+ this exists as a file in devtools. It needs to be run as apache (usually www-data), and an example (for a sudoer) command would be:

```
sudo -u www-data php doc/devtools/rebuild_search_index.php
```

In Tiki 9+ you can also now use:

rebuild from command line

```
sudo -u www-data php lib/search/shell.php rebuild
```

Get debug information during rebuild

```
sudo -u www-data php lib/search/shell.php rebuild log
```

The debug information is in file temp/Search_Indexer.log

Process part of the unindexed items queue (number to process is configurable)

```
sudo -u www-data php lib/search/shell.php process
```

If a rebuild is already ongoing, it will tell you:

Rebuild in progress

```
sudo -u www-data php lib/search/shell.php process
2012-04-13T10:44:37+00:00 INFO (6): Rebuild in progress - exiting.
```

If this is in error, maybe a rebuild was interrupted and you can't rebuild because it claims that a rebuild is ongoing, you need to do:

Recover from interrupted rebuid

```
sudo -u www-data rm -rf temp/unified-index-new/
```

Multilingual

For now the search has only an English Analyser

- to add another language
  - update ib/core/Search/Index/Lucene.php
  - add the appropriate files in lib/core/StandardAnalyser

You can customize the stop words in lib/core/StandardAnalyzer/Analyzer/Standard/lang.php

Tips on setting a new field

Each object to be indexed has fields that identify the tiki information and the way it is managed in the indexer.

Tiki types of field

type	tokenized	indexed	stored in index database	special	lucene type
plaintext	y	y	n		unstored
wikitext	y	y	n	parsed before indexed	unstored
timestamp	n	y	y		keyword
identifier	y	y	n		keyword????
multivalue	y	y	n		unstored
sortable	y	y	y		?????
text	y	y	y		stored

More information on Lucene type fields

Developer Notes

Content sources

These can be found in `lib/core/Search/ContentSource`

What determines if a field is full-text searchable in the default **contents** field

For each indexed source, there is a getGlobalFields() function. If the field in question is listed as one of the keys of the returned array, then it can be searched using full-text in the default **contents** fields, in addition, if the value is set to true, then it is preserved after it is included in the contents field, otherwise it is discarded. For example:

```
function getGlobalFields()
        {
                return array(
                        'title' => true,  // included in contents field and the field itself
is preserved
                        'description' => true,  // included in contents field and the field
itself is preserved

                        'article_content' => false,  // included in contents field and the
field itself is discarded thereafter
                        'article_topline' => false, // included in contents field and the
field itself is discarded thereafter
                        'article_subtitle' => false, // included in contents field and the
field itself is discarded thereafter
                        'article_author' => false, // included in contents field and the field
itself is discarded thereafter
                );
        }
```

Adding new fields to be indexed

- Remember to add it to the getProvidedFields functions. That function is used to get a list of indexed fields to iterate through for various purposes within Tiki.

Related pages

- Search-related wishes
- http://www.opensearch.org/
- TNTSearch
  - https://packagist.org/packages/teamtnt/tntsearch
  - https://www.openhub.net/p/tntsearch

Alias

- UnifiedIndex

- UnifiedSearch
- Unified Search