Contents of /trunk/phase3/maintenance/tables.sql

Revision **113110** - (**show annotations**) (**download**)
Tue Mar 6 00:09:18 2012 UTC (7 months, 1 week ago) by *reedy*
File size: 54092 byte(s)

Followup r113109, forward port the rest of r25267 for Bug 10788 - Filter page histories by user, or contributions by title

```
1  -- SQL to create the initial tables for the MediaWiki database.
2  -- This is read and executed by the install script; you should
3  -- not have to run it by itself unless doing a manual install.
4
5  -- This is a shared schema file used for both MySQL and SQLite installs.
6
7  --
8  -- General notes:
9  --
10 -- If possible, create tables as InnoDB to benefit from the
11 -- superior resiliency against crashes and ability to read
12 -- during writes (and write during reads!)
13 --
14 -- Only the 'searchindex' table requires MyISAM due to the
15 -- requirement for fulltext index support, which is missing
16 -- from InnoDB.
17 --
18 --
19 -- The MySQL table backend for MediaWiki currently uses
20 -- 14-character BINARY or VARBINARY fields to store timestamps.
21 -- The format is YYYYMMDDHHMMSS, which is derived from the
22 -- text format of MySQL's TIMESTAMP fields.
23 --
24 -- Historically TIMESTAMP fields were used, but abandoned
25 -- in early 2002 after a lot of trouble with the fields
26 -- auto-updating.
27 --
28 -- The Postgres backend uses DATETIME fields for timestamps,
29 -- and we will migrate the MySQL definitions at some point as
30 -- well.
31 --
32 --
33 -- The /*_*/ comments in this and other files are
34 -- replaced with the defined table prefix by the installer
35 -- and updater scripts. If you are installing or running
36 -- updates manually, you will need to manually insert the
37 -- table prefix if any when running these scripts.
38 --
39
40
41 --
42 -- The user table contains basic account information,
43 -- authentication keys, etc.
44 --
45 -- Some multi-wiki sites may share a single central user table
46 -- between separate wikis using the $wgSharedDB setting.
47 --
48 -- Note that when a external authentication plugin is used,
49 -- user table entries still need to be created to store
50 -- preferences and to key tracking information in the other
51 -- tables.
52 --
53 CREATE TABLE /*_*/user (
54   user_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
55
56   -- Usernames must be unique, must not be in the form of
57   -- an IP address. _Shouldn't_ allow slashes or case
58   -- conflicts. Spaces are allowed, and are _not_ converted
59   -- to underscores like titles. See the User::newFromName() for
60   -- the specific tests that usernames have to pass.
61   user_name varchar(255) binary NOT NULL default '',
62
63   -- Optional 'real name' to be displayed in credit listings
64   user_real_name varchar(255) binary NOT NULL default '',
65
66   -- Password hashes, see User::crypt() and User::comparePasswords()
```

```
  -- in User.php for the algorithm
  user_password tinyblob NOT NULL,

  -- When using 'mail me a new password', a random
  -- password is generated and the hash stored here.
  -- The previous password is left in place until
  -- someone actually logs in with the new password,
  -- at which point the hash is moved to user_password
  -- and the old password is invalidated.
  user_newpassword tinyblob NOT NULL,

  -- Timestamp of the last time when a new password was
  -- sent, for throttling and expiring purposes
  -- Emailed passwords will expire $wgNewPasswordExpiry
  -- (a week) after being set. If user_newpass_time is NULL
  -- (eg. created by mail) it doesn't expire.
  user_newpass_time binary(14),

  -- Note: email should be restricted, not public info.
  -- Same with passwords.
  user_email tinytext NOT NULL,

  -- This is a timestamp which is updated when a user
  -- logs in, logs out, changes preferences, or performs
  -- some other action requiring HTML cache invalidation
  -- to ensure that the UI is updated.
  user_touched binary(14) NOT NULL default '',

  -- A pseudorandomly generated value that is stored in
  -- a cookie when the "remember password" feature is
  -- used (previously, a hash of the password was used, but
  -- this was vulnerable to cookie-stealing attacks)
  user_token binary(32) NOT NULL default '',

  -- Initially NULL; when a user's e-mail address has been
  -- validated by returning with a mailed token, this is
  -- set to the current timestamp.
  user_email_authenticated binary(14),

  -- Randomly generated token created when the e-mail address
  -- is set and a confirmation test mail sent.
  user_email_token binary(32),

  -- Expiration date for the user_email_token
  user_email_token_expires binary(14),

  -- Timestamp of account registration.
  -- Accounts predating this schema addition may contain NULL.
  user_registration binary(14),

  -- Count of edits and edit-like actions.
  --
  -- *NOT* intended to be an accurate copy of COUNT(*) WHERE rev_user=user_id
  -- May contain NULL for old accounts if batch-update scripts haven't been
  -- run, as well as listing deleted edits and other myriad ways it could be
  -- out of sync.
  --
  -- Meant primarily for heuristic checks to give an impression of whether
  -- the account has been used much.
  --
  user_editcount int
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/user_name ON /*_*/user (user_name);
CREATE INDEX /*i*/user_email_token ON /*_*/user (user_email_token);
CREATE INDEX /*i*/user_email ON /*_*/user (user_email(50));


--
-- User permissions have been broken out to a separate table;
-- this allows sites with a shared user table to have different
-- permissions assigned to a user in each project.
--
-- This table replaces the old user_rights field which used a
-- comma-separated blob.
--
CREATE TABLE /*_*/user_groups (
  -- Key to user_id
```

```
  ug_user int unsigned NOT NULL default 0,

  -- Group names are short symbolic string keys.
  -- The set of group names is open-ended, though in practice
  -- only some predefined ones are likely to be used.
  --
  -- At runtime $wgGroupPermissions will associate group keys
  -- with particular permissions. A user will have the combined
  -- permissions of any group they're explicitly in, plus
  -- the implicit '*' and 'user' groups.
  ug_group varbinary(32) NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/ug_user_group ON /*_*/user_groups (ug_user,ug_group);
CREATE INDEX /*i*/ug_group ON /*_*/user_groups (ug_group);

-- Stores the groups the user has once belonged to.
-- The user may still belong to these groups (check user_groups).
-- Users are not autopromoted to groups from which they were removed.
CREATE TABLE /*_*/user_former_groups (
  -- Key to user_id
  ufg_user int unsigned NOT NULL default 0,
  ufg_group varbinary(32) NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/ufg_user_group ON /*_*/user_former_groups (ufg_user,ufg_group);

--
-- Stores notifications of user talk page changes, for the display
-- of the "you have new messages" box
--
CREATE TABLE /*_*/user_newtalk (
  -- Key to user.user_id
  user_id int NOT NULL default 0,
  -- If the user is an anonymous user their IP address is stored here
  -- since the user_id of 0 is ambiguous
  user_ip varbinary(40) NOT NULL default '',
  -- The highest timestamp of revisions of the talk page viewed
  -- by this user
  user_last_timestamp varbinary(14) NULL default NULL
) /*$wgDBTableOptions*/;

-- Indexes renamed for SQLite in 1.14
CREATE INDEX /*i*/un_user_id ON /*_*/user_newtalk (user_id);
CREATE INDEX /*i*/un_user_ip ON /*_*/user_newtalk (user_ip);


--
-- User preferences and perhaps other fun stuff. :)
-- Replaces the old user.user_options blob, with a couple nice properties:
--
-- 1) We only store non-default settings, so changes to the defauls
--    are now reflected for everybody, not just new accounts.
-- 2) We can more easily do bulk lookups, statistics, or modifications of
--    saved options since it's a sane table structure.
--
CREATE TABLE /*_*/user_properties (
  -- Foreign key to user.user_id
  up_user int NOT NULL,

  -- Name of the option being saved. This is indexed for bulk lookup.
  up_property varbinary(255) NOT NULL,

  -- Property value as a string.
  up_value blob
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/user_properties_user_property ON /*_*/user_properties (up_user,up_property);
CREATE INDEX /*i*/user_properties_property ON /*_*/user_properties (up_property);

--
-- Core of the wiki: each page has an entry here which identifies
-- it by title and contains some essential metadata.
--
CREATE TABLE /*_*/page (
  -- Unique identifier number. The page_id will be preserved across
  -- edits and rename operations, but not deletions and recreations.
  page_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
223
224   -- A page name is broken into a namespace and a title.
225   -- The namespace keys are UI-language-independent constants,
226   -- defined in includes/Defines.php
227   page_namespace int NOT NULL,
228
229   -- The rest of the title, as text.
230   -- Spaces are transformed into underscores in title storage.
231   page_title varchar(255) binary NOT NULL,
232
233   -- Comma-separated set of permission keys indicating who
234   -- can move or edit the page.
235   page_restrictions tinyblob NOT NULL,
236
237   -- Number of times this page has been viewed.
238   page_counter bigint unsigned NOT NULL default 0,
239
240   -- 1 indicates the article is a redirect.
241   page_is_redirect tinyint unsigned NOT NULL default 0,
242
243   -- 1 indicates this is a new entry, with only one edit.
244   -- Not all pages with one edit are new pages.
245   page_is_new tinyint unsigned NOT NULL default 0,
246
247   -- Random value between 0 and 1, used for Special:Randompage
248   page_random real unsigned NOT NULL,
249
250   -- This timestamp is updated whenever the page changes in
251   -- a way requiring it to be re-rendered, invalidating caches.
252   -- Aside from editing this includes permission changes,
253   -- creation or deletion of linked pages, and alteration
254   -- of contained templates.
255   page_touched binary(14) NOT NULL default '',
256
257   -- Handy key to revision.rev_id of the current revision.
258   -- This may be 0 during page creation, but that shouldn't
259   -- happen outside of a transaction... hopefully.
260   page_latest int unsigned NOT NULL,
261
262   -- Uncompressed length in bytes of the page's current source text.
263   page_len int unsigned NOT NULL
264 ) /*$wgDBTableOptions*/;
265
266 CREATE UNIQUE INDEX /*i*/name_title ON /*_*/page (page_namespace,page_title);
267 CREATE INDEX /*i*/page_random ON /*_*/page (page_random);
268 CREATE INDEX /*i*/page_len ON /*_*/page (page_len);
269 CREATE INDEX /*i*/page_redirect_namespace_len ON /*_*/page (page_is_redirect, page_namespace, page_len);
270
271 --
272 -- Every edit of a page creates also a revision row.
273 -- This stores metadata about the revision, and a reference
274 -- to the text storage backend.
275 --
276 CREATE TABLE /*_*/revision (
277   -- Unique ID to identify each revision
278   rev_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
279
280   -- Key to page_id. This should _never_ be invalid.
281   rev_page int unsigned NOT NULL,
282
283   -- Key to text.old_id, where the actual bulk text is stored.
284   -- It's possible for multiple revisions to use the same text,
285   -- for instance revisions where only metadata is altered
286   -- or a rollback to a previous version.
287   rev_text_id int unsigned NOT NULL,
288
289   -- Text comment summarizing the change.
290   -- This text is shown in the history and other changes lists,
291   -- rendered in a subset of wiki markup by Linker::formatComment()
292   rev_comment tinyblob NOT NULL,
293
294   -- Key to user.user_id of the user who made this edit.
295   -- Stores 0 for anonymous edits and for some mass imports.
296   rev_user int unsigned NOT NULL default 0,
297
298   -- Text username or IP address of the editor.
299   rev_user_text varchar(255) binary NOT NULL default '',
300
```

```
301   -- Timestamp of when revision was created
302   rev_timestamp binary(14) NOT NULL default '',
303
304   -- Records whether the user marked the 'minor edit' checkbox.
305   -- Many automated edits are marked as minor.
306   rev_minor_edit tinyint unsigned NOT NULL default 0,
307
308   -- Restrictions on who can access this revision
309   rev_deleted tinyint unsigned NOT NULL default 0,
310
311   -- Length of this revision in bytes
312   rev_len int unsigned,
313
314   -- Key to revision.rev_id
315   -- This field is used to add support for a tree structure (The Adjacency List Model)
316   rev_parent_id int unsigned default NULL,
317
318   -- SHA-1 text content hash in base-36
319   rev_sha1 varbinary(32) NOT NULL default ''
320
321 ) /*$wgDBTableOptions*/ MAX_ROWS=10000000 AVG_ROW_LENGTH=1024;
322 -- In case tables are created as MyISAM, use row hints for MySQL <5.0 to avoid 4GB limit
323
324 CREATE UNIQUE INDEX /*i*/rev_page_id ON /*_*/revision (rev_page, rev_id);
325 CREATE INDEX /*i*/rev_timestamp ON /*_*/revision (rev_timestamp);
326 CREATE INDEX /*i*/page_timestamp ON /*_*/revision (rev_page,rev_timestamp);
327 CREATE INDEX /*i*/user_timestamp ON /*_*/revision (rev_user,rev_timestamp);
328 CREATE INDEX /*i*/usertext_timestamp ON /*_*/revision (rev_user_text,rev_timestamp);
329 CREATE INDEX /*i*/page_user_timestamp ON /*_*/revision  (rev_page,rev_user,rev_timestamp);
330
331 --
332 -- Holds text of individual page revisions.
333 --
334 -- Field names are a holdover from the 'old' revisions table in
335 -- MediaWiki 1.4 and earlier: an upgrade will transform that
336 -- table into the 'text' table to minimize unnecessary churning
337 -- and downtime. If upgrading, the other fields will be left unused.
338 --
339 CREATE TABLE /*_*/text (
340   -- Unique text storage key number.
341   -- Note that the 'oldid' parameter used in URLs does *not*
342   -- refer to this number anymore, but to rev_id.
343   --
344   -- revision.rev_text_id is a key to this column
345   old_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
346
347   -- Depending on the contents of the old_flags field, the text
348   -- may be convenient plain text, or it may be funkily encoded.
349   old_text mediumblob NOT NULL,
350
351   -- Comma-separated list of flags:
352   -- gzip: text is compressed with PHP's gzdeflate() function.
353   -- utf8: text was stored as UTF-8.
354   --       If $wgLegacyEncoding option is on, rows *without* this flag
355   --       will be converted to UTF-8 transparently at load time.
356   -- object: text field contained a serialized PHP object.
357   --         The object either contains multiple versions compressed
358   --         together to achieve a better compression ratio, or it refers
359   --         to another row where the text can be found.
360   old_flags tinyblob NOT NULL
361 ) /*$wgDBTableOptions*/ MAX_ROWS=10000000 AVG_ROW_LENGTH=10240;
362 -- In case tables are created as MyISAM, use row hints for MySQL <5.0 to avoid 4GB limit
363
364
365 --
366 -- Holding area for deleted articles, which may be viewed
367 -- or restored by admins through the Special:Undelete interface.
368 -- The fields generally correspond to the page, revision, and text
369 -- fields, with several caveats.
370 --
371 CREATE TABLE /*_*/archive (
372   ar_namespace int NOT NULL default 0,
373   ar_title varchar(255) binary NOT NULL default '',
374
375   -- Newly deleted pages will not store text in this table,
376   -- but will reference the separately existing text rows.
377   -- This field is retained for backwards compatibility,
378   -- so old archived pages will remain accessible after
```

```
  -- upgrading from 1.4 to 1.5.
  -- Text may be gzipped or otherwise funky.
  ar_text mediumblob NOT NULL,

  -- Basic revision stuff...
  ar_comment tinyblob NOT NULL,
  ar_user int unsigned NOT NULL default 0,
  ar_user_text varchar(255) binary NOT NULL,
  ar_timestamp binary(14) NOT NULL default '',
  ar_minor_edit tinyint NOT NULL default 0,

  -- See ar_text note.
  ar_flags tinyblob NOT NULL,

  -- When revisions are deleted, their unique rev_id is stored
  -- here so it can be retained after undeletion. This is necessary
  -- to retain permalinks to given revisions after accidental delete
  -- cycles or messy operations like history merges.
  --
  -- Old entries from 1.4 will be NULL here, and a new rev_id will
  -- be created on undeletion for those revisions.
  ar_rev_id int unsigned,

  -- For newly deleted revisions, this is the text.old_id key to the
  -- actual stored text. To avoid breaking the block-compression scheme
  -- and otherwise making storage changes harder, the actual text is
  -- *not* deleted from the text table, merely hidden by removal of the
  -- page and revision entries.
  --
  -- Old entries deleted under 1.2-1.4 will have NULL here, and their
  -- ar_text and ar_flags fields will be used to create a new text
  -- row upon undeletion.
  ar_text_id int unsigned,

  -- rev_deleted for archives
  ar_deleted tinyint unsigned NOT NULL default 0,

  -- Length of this revision in bytes
  ar_len int unsigned,

  -- Reference to page_id. Useful for sysadmin fixing of large pages
  -- merged together in the archives, or for cleanly restoring a page
  -- at its original ID number if possible.
  --
  -- Will be NULL for pages deleted prior to 1.11.
  ar_page_id int unsigned,

  -- Original previous revision
  ar_parent_id int unsigned default NULL,

  -- SHA-1 text content hash in base-36
  ar_sha1 varbinary(32) NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE INDEX /*i*/name_title_timestamp ON /*_*/archive (ar_namespace,ar_title,ar_timestamp);
CREATE INDEX /*i*/ar_usertext_timestamp ON /*_*/archive (ar_user_text,ar_timestamp);
CREATE INDEX /*i*/ar_revid ON /*_*/archive (ar_rev_id);



--
-- Track page-to-page hyperlinks within the wiki.
--
CREATE TABLE /*_*/pagelinks (
  -- Key to the page_id of the page containing the link.
  pl_from int unsigned NOT NULL default 0,

  -- Key to page_namespace/page_title of the target page.
  -- The target page may or may not exist, and due to renames
  -- and deletions may refer to different page records as time
  -- goes by.
  pl_namespace int NOT NULL default 0,
  pl_title varchar(255) binary NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/pl_from ON /*_*/pagelinks (pl_from,pl_namespace,pl_title);
CREATE UNIQUE INDEX /*i*/pl_namespace ON /*_*/pagelinks (pl_namespace,pl_title,pl_from);


```

```
--
-- Track template inclusions.
--
CREATE TABLE /*_*/templatelinks (
  -- Key to the page_id of the page containing the link.
  tl_from int unsigned NOT NULL default 0,

  -- Key to page_namespace/page_title of the target page.
  -- The target page may or may not exist, and due to renames
  -- and deletions may refer to different page records as time
  -- goes by.
  tl_namespace int NOT NULL default 0,
  tl_title varchar(255) binary NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/tl_from ON /*_*/templatelinks (tl_from,tl_namespace,tl_title);
CREATE UNIQUE INDEX /*i*/tl_namespace ON /*_*/templatelinks (tl_namespace,tl_title,tl_from);



--
-- Track links to images *used inline*
-- We don't distinguish live from broken links here, so
-- they do not need to be changed on upload/removal.
--
CREATE TABLE /*_*/imagelinks (
  -- Key to page_id of the page containing the image / media link.
  il_from int unsigned NOT NULL default 0,

  -- Filename of target image.
  -- This is also the page_title of the file's description page;
  -- all such pages are in namespace 6 (NS_FILE).
  il_to varchar(255) binary NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/il_from ON /*_*/imagelinks (il_from,il_to);
CREATE UNIQUE INDEX /*i*/il_to ON /*_*/imagelinks (il_to,il_from);



--
-- Track category inclusions *used inline*
-- This tracks a single level of category membership
--
CREATE TABLE /*_*/categorylinks (
  -- Key to page_id of the page defined as a category member.
  cl_from int unsigned NOT NULL default 0,

  -- Name of the category.
  -- This is also the page_title of the category's description page;
  -- all such pages are in namespace 14 (NS_CATEGORY).
  cl_to varchar(255) binary NOT NULL default '',

  -- A binary string obtained by applying a sortkey generation algorithm
  -- (Collation::getSortKey()) to page_title, or cl_sortkey_prefix . "\n"
  -- . page_title if cl_sortkey_prefix is nonempty.
  cl_sortkey varbinary(230) NOT NULL default '',

  -- A prefix for the raw sortkey manually specified by the user, either via
  -- [[Category:Foo|prefix]] or {{defaultsort:prefix}}.  If nonempty, it's
  -- concatenated with a line break followed by the page title before the sortkey
  -- conversion algorithm is run.  We store this so that we can update
  -- collations without reparsing all pages.
  -- Note: If you change the length of this field, you also need to change
  -- code in LinksUpdate.php. See bug 25254.
  cl_sortkey_prefix varchar(255) binary NOT NULL default '',

  -- This isn't really used at present. Provided for an optional
  -- sorting method by approximate addition time.
  cl_timestamp timestamp NOT NULL,

  -- Stores $wgCategoryCollation at the time cl_sortkey was generated.  This
  -- can be used to install new collation versions, tracking which rows are not
  -- yet updated.  '' means no collation, this is a legacy row that needs to be
  -- updated by updateCollation.php.  In the future, it might be possible to
  -- specify different collations per category.
  cl_collation varbinary(32) NOT NULL default '',

  -- Stores whether cl_from is a category, file, or other page, so we can
  -- paginate the three categories separately.  This never has to be updated
```

```
  -- after the page is created, since none of these page types can be moved to
  -- any other.
  cl_type ENUM('page', 'subcat', 'file') NOT NULL default 'page'
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/cl_from ON /*_*/categorylinks (cl_from,cl_to);

-- We always sort within a given category, and within a given type.  FIXME:
-- Formerly this index didn't cover cl_type (since that didn't exist), so old
-- callers won't be using an index: fix this?
CREATE INDEX /*i*/cl_sortkey ON /*_*/categorylinks (cl_to,cl_type,cl_sortkey,cl_from);

-- Not really used?
CREATE INDEX /*i*/cl_timestamp ON /*_*/categorylinks (cl_to,cl_timestamp);

-- For finding rows with outdated collation
CREATE INDEX /*i*/cl_collation ON /*_*/categorylinks (cl_collation);

--
-- Track all existing categories.  Something is a category if 1) it has an en-
-- try somewhere in categorylinks, or 2) it once did.  Categories might not
-- have corresponding pages, so they need to be tracked separately.
--
CREATE TABLE /*_*/category (
  -- Primary key
  cat_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,

  -- Name of the category, in the same form as page_title (with underscores).
  -- If there is a category page corresponding to this category, by definition,
  -- it has this name (in the Category namespace).
  cat_title varchar(255) binary NOT NULL,

  -- The numbers of member pages (including categories and media), subcatego-
  -- ries, and Image: namespace members, respectively.  These are signed to
  -- make underflow more obvious.  We make the first number include the second
  -- two for better sorting: subtracting for display is easy, adding for order-
  -- ing is not.
  cat_pages int signed NOT NULL default 0,
  cat_subcats int signed NOT NULL default 0,
  cat_files int signed NOT NULL default 0,

  -- Reserved for future use
  cat_hidden tinyint unsigned NOT NULL default 0
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/cat_title ON /*_*/category (cat_title);

-- For Special:Mostlinkedcategories
CREATE INDEX /*i*/cat_pages ON /*_*/category (cat_pages);



--
-- Track links to external URLs
--
CREATE TABLE /*_*/externallinks (
  -- page_id of the referring page
  el_from int unsigned NOT NULL default 0,

  -- The URL
  el_to blob NOT NULL,

  -- In the case of HTTP URLs, this is the URL with any username or password
  -- removed, and with the labels in the hostname reversed and converted to
  -- lower case. An extra dot is added to allow for matching of either
  -- example.com or *.example.com in a single scan.
  -- Example:
  --      http://user:password@sub.example.com/page.html
  --   becomes
  --      http://com.example.sub./page.html
  -- which allows for fast searching for all pages under example.com with the
  -- clause:
  --      WHERE el_index LIKE 'http://com.example.%'
  el_index blob NOT NULL
) /*$wgDBTableOptions*/;

CREATE INDEX /*i*/el_from ON /*_*/externallinks (el_from, el_to(40));
CREATE INDEX /*i*/el_to ON /*_*/externallinks (el_to(60), el_from);
CREATE INDEX /*i*/el_index ON /*_*/externallinks (el_index(60));
```

```
613
614
615 --
616 -- Track external user accounts, if ExternalAuth is used
617 --
618 CREATE TABLE /*_*/external_user (
619   -- Foreign key to user_id
620   eu_local_id int unsigned NOT NULL PRIMARY KEY,
621
622   -- Some opaque identifier provided by the external database
623   eu_external_id varchar(255) binary NOT NULL
624 ) /*$wgDBTableOptions*/;
625
626 CREATE UNIQUE INDEX /*i*/eu_external_id ON /*_*/external_user (eu_external_id);
627
628
629 --
630 -- Track interlanguage links
631 --
632 CREATE TABLE /*_*/langlinks (
633   -- page_id of the referring page
634   ll_from int unsigned NOT NULL default 0,
635
636   -- Language code of the target
637   ll_lang varbinary(20) NOT NULL default '',
638
639   -- Title of the target, including namespace
640   ll_title varchar(255) binary NOT NULL default ''
641 ) /*$wgDBTableOptions*/;
642
643 CREATE UNIQUE INDEX /*i*/ll_from ON /*_*/langlinks (ll_from, ll_lang);
644 CREATE INDEX /*i*/ll_lang ON /*_*/langlinks (ll_lang, ll_title);
645
646
647 --
648 -- Track inline interwiki links
649 --
650 CREATE TABLE /*_*/iwlinks (
651   -- page_id of the referring page
652   iwl_from int unsigned NOT NULL default 0,
653
654   -- Interwiki prefix code of the target
655   iwl_prefix varbinary(20) NOT NULL default '',
656
657   -- Title of the target, including namespace
658   iwl_title varchar(255) binary NOT NULL default ''
659 ) /*$wgDBTableOptions*/;
660
661 CREATE UNIQUE INDEX /*i*/iwl_from ON /*_*/iwlinks (iwl_from, iwl_prefix, iwl_title);
662 CREATE UNIQUE INDEX /*i*/iwl_prefix_title_from ON /*_*/iwlinks (iwl_prefix, iwl_title, iwl_from);
663
664
665 --
666 -- Contains a single row with some aggregate info
667 -- on the state of the site.
668 --
669 CREATE TABLE /*_*/site_stats (
670   -- The single row should contain 1 here.
671   ss_row_id int unsigned NOT NULL,
672
673   -- Total number of page views, if hit counters are enabled.
674   ss_total_views bigint unsigned default 0,
675
676   -- Total number of edits performed.
677   ss_total_edits bigint unsigned default 0,
678
679   -- An approximate count of pages matching the following criteria:
680   -- * in namespace 0
681   -- * not a redirect
682   -- * contains the text '[['
683   -- See Article::isCountable() in includes/Article.php
684   ss_good_articles bigint unsigned default 0,
685
686   -- Total pages, theoretically equal to SELECT COUNT(*) FROM page; except faster
687   ss_total_pages bigint default '-1',
688
689   -- Number of users, theoretically equal to SELECT COUNT(*) FROM user;
690   ss_users bigint default '-1',
```

```
691
692   -- Number of users that still edit
693   ss_active_users bigint default '-1',
694
695   -- Deprecated, no longer updated as of 1.5
696   ss_admins int default '-1',
697
698   -- Number of images, equivalent to SELECT COUNT(*) FROM image
699   ss_images int default 0
700 ) /*$wgDBTableOptions*/;
701
702 -- Pointless index to assuage developer superstitions
703 CREATE UNIQUE INDEX /*i*/ss_row_id ON /*_*/site_stats (ss_row_id);
704
705
706 --
707 -- Stores an ID for every time any article is visited;
708 -- depending on $wgHitcounterUpdateFreq, it is
709 -- periodically cleared and the page_counter column
710 -- in the page table updated for all the articles
711 -- that have been visited.)
712 --
713 CREATE TABLE /*_*/hitcounter (
714   hc_id int unsigned NOT NULL
715 ) ENGINE=HEAP MAX_ROWS=25000;
716
717
718 --
719 -- The internet is full of jerks, alas. Sometimes it's handy
720 -- to block a vandal or troll account.
721 --
722 CREATE TABLE /*_*/ipblocks (
723   -- Primary key, introduced for privacy.
724   ipb_id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
725
726   -- Blocked IP address in dotted-quad form or user name.
727   ipb_address tinyblob NOT NULL,
728
729   -- Blocked user ID or 0 for IP blocks.
730   ipb_user int unsigned NOT NULL default 0,
731
732   -- User ID who made the block.
733   ipb_by int unsigned NOT NULL default 0,
734
735   -- User name of blocker
736   ipb_by_text varchar(255) binary NOT NULL default '',
737
738   -- Text comment made by blocker.
739   ipb_reason tinyblob NOT NULL,
740
741   -- Creation (or refresh) date in standard YMDHMS form.
742   -- IP blocks expire automatically.
743   ipb_timestamp binary(14) NOT NULL default '',
744
745   -- Indicates that the IP address was banned because a banned
746   -- user accessed a page through it. If this is 1, ipb_address
747   -- will be hidden, and the block identified by block ID number.
748   ipb_auto bool NOT NULL default 0,
749
750   -- If set to 1, block applies only to logged-out users
751   ipb_anon_only bool NOT NULL default 0,
752
753   -- Block prevents account creation from matching IP addresses
754   ipb_create_account bool NOT NULL default 1,
755
756   -- Block triggers autoblocks
757   ipb_enable_autoblock bool NOT NULL default '1',
758
759   -- Time at which the block will expire.
760   -- May be "infinity"
761   ipb_expiry varbinary(14) NOT NULL default '',
762
763   -- Start and end of an address range, in hexadecimal
764   -- Size chosen to allow IPv6
765   ipb_range_start tinyblob NOT NULL,
766   ipb_range_end tinyblob NOT NULL,
767
768   -- Flag for entries hidden from users and Sysops
```

```
  ipb_deleted bool NOT NULL default 0,

  -- Block prevents user from accessing Special:Emailuser
  ipb_block_email bool NOT NULL default 0,

  -- Block allows user to edit their own talk page
  ipb_allow_usertalk bool NOT NULL default 0

) /*$wgDBTableOptions*/;

-- Unique index to support "user already blocked" messages
-- Any new options which prevent collisions should be included
CREATE UNIQUE INDEX /*i*/ipb_address ON /*_*/ipblocks (ipb_address(255), ipb_user, ipb_auto, ipb_anon_only);

CREATE INDEX /*i*/ipb_user ON /*_*/ipblocks (ipb_user);
CREATE INDEX /*i*/ipb_range ON /*_*/ipblocks (ipb_range_start(8), ipb_range_end(8));
CREATE INDEX /*i*/ipb_timestamp ON /*_*/ipblocks (ipb_timestamp);
CREATE INDEX /*i*/ipb_expiry ON /*_*/ipblocks (ipb_expiry);



--
-- Uploaded images and other files.
--
CREATE TABLE /*_*/image (
  -- Filename.
  -- This is also the title of the associated description page,
  -- which will be in namespace 6 (NS_FILE).
  img_name varchar(255) binary NOT NULL default '' PRIMARY KEY,

  -- File size in bytes.
  img_size int unsigned NOT NULL default 0,

  -- For images, size in pixels.
  img_width int NOT NULL default 0,
  img_height int NOT NULL default 0,

  -- Extracted EXIF metadata stored as a serialized PHP array.
  img_metadata mediumblob NOT NULL,

  -- For images, bits per pixel if known.
  img_bits int NOT NULL default 0,

  -- Media type as defined by the MEDIATYPE_xxx constants
  img_media_type ENUM("UNKNOWN", "BITMAP", "DRAWING", "AUDIO", "VIDEO", "MULTIMEDIA", "OFFICE", "TEXT", "EXECUTABLE", "ARCHIVE") default NULL,

  -- major part of a MIME media type as defined by IANA
  -- see http://www.iana.org/assignments/media-types/
  img_major_mime ENUM("unknown", "application", "audio", "image", "text", "video", "message", "model", "multipart") NOT NULL default "unknown",

  -- minor part of a MIME media type as defined by IANA
  -- the minor parts are not required to adher to any standard
  -- but should be consistent throughout the database
  -- see http://www.iana.org/assignments/media-types/
  img_minor_mime varbinary(100) NOT NULL default "unknown",

  -- Description field as entered by the uploader.
  -- This is displayed in image upload history and logs.
  img_description tinyblob NOT NULL,

  -- user_id and user_name of uploader.
  img_user int unsigned NOT NULL default 0,
  img_user_text varchar(255) binary NOT NULL,

  -- Time of the upload.
  img_timestamp varbinary(14) NOT NULL default '',

  -- SHA-1 content hash in base-36
  img_sha1 varbinary(32) NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE INDEX /*i*/img_usertext_timestamp ON /*_*/image (img_user_text,img_timestamp);
-- Used by Special:ListFiles for sort-by-size
CREATE INDEX /*i*/img_size ON /*_*/image (img_size);
-- Used by Special:Newimages and Special:ListFiles
CREATE INDEX /*i*/img_timestamp ON /*_*/image (img_timestamp);
-- Used in API and duplicate search
CREATE INDEX /*i*/img_sha1 ON /*_*/image (img_sha1);

```

```
847
848 --
849 -- Previous revisions of uploaded files.
850 -- Awkwardly, image rows have to be moved into
851 -- this table at re-upload time.
852 --
853 CREATE TABLE /*_*/oldimage (
854   -- Base filename: key to image.img_name
855   oi_name varchar(255) binary NOT NULL default '',
856
857   -- Filename of the archived file.
858   -- This is generally a timestamp and '!' prepended to the base name.
859   oi_archive_name varchar(255) binary NOT NULL default '',
860
861   -- Other fields as in image...
862   oi_size int unsigned NOT NULL default 0,
863   oi_width int NOT NULL default 0,
864   oi_height int NOT NULL default 0,
865   oi_bits int NOT NULL default 0,
866   oi_description tinyblob NOT NULL,
867   oi_user int unsigned NOT NULL default 0,
868   oi_user_text varchar(255) binary NOT NULL,
869   oi_timestamp binary(14) NOT NULL default '',
870
871   oi_metadata mediumblob NOT NULL,
872   oi_media_type ENUM("UNKNOWN", "BITMAP", "DRAWING", "AUDIO", "VIDEO", "MULTIMEDIA", "OFFICE", "TEXT", "EXECUTABLE", "ARCHIVE") default NULL,
873   oi_major_mime ENUM("unknown", "application", "audio", "image", "text", "video", "message", "model", "multipart") NOT NULL default "unknown",
874   oi_minor_mime varbinary(100) NOT NULL default "unknown",
875   oi_deleted tinyint unsigned NOT NULL default 0,
876   oi_sha1 varbinary(32) NOT NULL default ''
877 ) /*$wgDBTableOptions*/;
878
879 CREATE INDEX /*i*/oi_usertext_timestamp ON /*_*/oldimage (oi_user_text,oi_timestamp);
880 CREATE INDEX /*i*/oi_name_timestamp ON /*_*/oldimage (oi_name,oi_timestamp);
881 -- oi_archive_name truncated to 14 to avoid key length overflow
882 CREATE INDEX /*i*/oi_name_archive_name ON /*_*/oldimage (oi_name,oi_archive_name(14));
883 CREATE INDEX /*i*/oi_sha1 ON /*_*/oldimage (oi_sha1);
884
885
886 --
887 -- Record of deleted file data
888 --
889 CREATE TABLE /*_*/filearchive (
890   -- Unique row id
891   fa_id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
892
893   -- Original base filename; key to image.img_name, page.page_title, etc
894   fa_name varchar(255) binary NOT NULL default '',
895
896   -- Filename of archived file, if an old revision
897   fa_archive_name varchar(255) binary default '',
898
899   -- Which storage bin (directory tree or object store) the file data
900   -- is stored in. Should be 'deleted' for files that have been deleted;
901   -- any other bin is not yet in use.
902   fa_storage_group varbinary(16),
903
904   -- SHA-1 of the file contents plus extension, used as a key for storage.
905   -- eg 8f8a562add37052a1848ff7771a2c515db94baa9.jpg
906   --
907   -- If NULL, the file was missing at deletion time or has been purged
908   -- from the archival storage.
909   fa_storage_key varbinary(64) default '',
910
911   -- Deletion information, if this file is deleted.
912   fa_deleted_user int,
913   fa_deleted_timestamp binary(14) default '',
914   fa_deleted_reason text,
915
916   -- Duped fields from image
917   fa_size int unsigned default 0,
918   fa_width int default 0,
919   fa_height int default 0,
920   fa_metadata mediumblob,
921   fa_bits int default 0,
922   fa_media_type ENUM("UNKNOWN", "BITMAP", "DRAWING", "AUDIO", "VIDEO", "MULTIMEDIA", "OFFICE", "TEXT", "EXECUTABLE", "ARCHIVE") default NULL,
923   fa_major_mime ENUM("unknown", "application", "audio", "image", "text", "video", "message", "model", "multipart") default "unknown",
924   fa_minor_mime varbinary(100) default "unknown",
```

```
  fa_description tinyblob,
  fa_user int unsigned default 0,
  fa_user_text varchar(255) binary,
  fa_timestamp binary(14) default '',

  -- Visibility of deleted revisions, bitfield
  fa_deleted tinyint unsigned NOT NULL default 0
) /*$wgDBTableOptions*/;

-- pick out by image name
CREATE INDEX /*i*/fa_name ON /*_*/filearchive (fa_name, fa_timestamp);
-- pick out dupe files
CREATE INDEX /*i*/fa_storage_group ON /*_*/filearchive (fa_storage_group, fa_storage_key);
-- sort by deletion time
CREATE INDEX /*i*/fa_deleted_timestamp ON /*_*/filearchive (fa_deleted_timestamp);
-- sort by uploader
CREATE INDEX /*i*/fa_user_timestamp ON /*_*/filearchive (fa_user_text,fa_timestamp);


--
-- Store information about newly uploaded files before they're
-- moved into the actual filestore
--
CREATE TABLE /*_*/uploadstash (
        us_id int unsigned NOT NULL PRIMARY KEY auto_increment,

        -- the user who uploaded the file.
        us_user int unsigned NOT NULL,

        -- file key. this is how applications actually search for the file.
        -- this might go away, or become the primary key.
        us_key varchar(255) NOT NULL,

        -- the original path
        us_orig_path varchar(255) NOT NULL,

        -- the temporary path at which the file is actually stored
        us_path varchar(255) NOT NULL,

        -- which type of upload the file came from (sometimes)
        us_source_type varchar(50),

        -- the date/time on which the file was added
        us_timestamp varbinary(14) not null,

        us_status varchar(50) not null,

        -- chunk counter starts at 0, current offset is stored in us_size
        us_chunk_inx int unsigned NULL,

        -- file properties from File::getPropsFromPath.  these may prove unnecessary.
        --
        us_size int unsigned NOT NULL,
        -- this hash comes from File::sha1Base36(), and is 31 characters
        us_sha1 varchar(31) NOT NULL,
        us_mime varchar(255),
        -- Media type as defined by the MEDIATYPE_xxx constants, should duplicate definition in the image table
        us_media_type ENUM("UNKNOWN", "BITMAP", "DRAWING", "AUDIO", "VIDEO", "MULTIMEDIA", "OFFICE", "TEXT", "EXECUTABLE", "ARCHIVE") default NULL,
        -- image-specific properties
        us_image_width int unsigned,
        us_image_height int unsigned,
        us_image_bits smallint unsigned

) /*$wgDBTableOptions*/;

-- sometimes there's a delete for all of a user's stuff.
CREATE INDEX /*i*/us_user ON /*_*/uploadstash (us_user);
-- pick out files by key, enforce key uniqueness
CREATE UNIQUE INDEX /*i*/us_key ON /*_*/uploadstash (us_key);
-- the abandoned upload cleanup script needs this
CREATE INDEX /*i*/us_timestamp ON /*_*/uploadstash (us_timestamp);


--
-- Primarily a summary table for Special:Recentchanges,
-- this table contains some additional info on edits from
-- the last few days, see Article::editUpdates()
--
```

```
1003 CREATE TABLE /*_*/recentchanges (
1004   rc_id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
1005   rc_timestamp varbinary(14) NOT NULL default '',
1006
1007   -- This is no longer used
1008   rc_cur_time varbinary(14) NOT NULL default '',
1009
1010   -- As in revision
1011   rc_user int unsigned NOT NULL default 0,
1012   rc_user_text varchar(255) binary NOT NULL,
1013
1014   -- When pages are renamed, their RC entries do _not_ change.
1015   rc_namespace int NOT NULL default 0,
1016   rc_title varchar(255) binary NOT NULL default '',
1017
1018   -- as in revision...
1019   rc_comment varchar(255) binary NOT NULL default '',
1020   rc_minor tinyint unsigned NOT NULL default 0,
1021
1022   -- Edits by user accounts with the 'bot' rights key are
1023   -- marked with a 1 here, and will be hidden from the
1024   -- default view.
1025   rc_bot tinyint unsigned NOT NULL default 0,
1026
1027   -- Set if this change corresponds to a page creation
1028   rc_new tinyint unsigned NOT NULL default 0,
1029
1030   -- Key to page_id (was cur_id prior to 1.5).
1031   -- This will keep links working after moves while
1032   -- retaining the at-the-time name in the changes list.
1033   rc_cur_id int unsigned NOT NULL default 0,
1034
1035   -- rev_id of the given revision
1036   rc_this_oldid int unsigned NOT NULL default 0,
1037
1038   -- rev_id of the prior revision, for generating diff links.
1039   rc_last_oldid int unsigned NOT NULL default 0,
1040
1041   -- The type of change entry (RC_EDIT,RC_NEW,RC_LOG)
1042   rc_type tinyint unsigned NOT NULL default 0,
1043
1044   -- These may no longer be used, with the new move log.
1045   rc_moved_to_ns tinyint unsigned NOT NULL default 0,
1046   rc_moved_to_title varchar(255) binary NOT NULL default '',
1047
1048   -- If the Recent Changes Patrol option is enabled,
1049   -- users may mark edits as having been reviewed to
1050   -- remove a warning flag on the RC list.
1051   -- A value of 1 indicates the page has been reviewed.
1052   rc_patrolled tinyint unsigned NOT NULL default 0,
1053
1054   -- Recorded IP address the edit was made from, if the
1055   -- $wgPutIPinRC option is enabled.
1056   rc_ip varbinary(40) NOT NULL default '',
1057
1058   -- Text length in characters before
1059   -- and after the edit
1060   rc_old_len int,
1061   rc_new_len int,
1062
1063   -- Visibility of recent changes items, bitfield
1064   rc_deleted tinyint unsigned NOT NULL default 0,
1065
1066   -- Value corresonding to log_id, specific log entries
1067   rc_logid int unsigned NOT NULL default 0,
1068   -- Store log type info here, or null
1069   rc_log_type varbinary(255) NULL default NULL,
1070   -- Store log action or null
1071   rc_log_action varbinary(255) NULL default NULL,
1072   -- Log params
1073   rc_params blob NULL
1074 ) /*$wgDBTableOptions*/;
1075
1076 CREATE INDEX /*i*/rc_timestamp ON /*_*/recentchanges (rc_timestamp);
1077 CREATE INDEX /*i*/rc_namespace_title ON /*_*/recentchanges (rc_namespace, rc_title);
1078 CREATE INDEX /*i*/rc_cur_id ON /*_*/recentchanges (rc_cur_id);
1079 CREATE INDEX /*i*/new_name_timestamp ON /*_*/recentchanges (rc_new,rc_namespace,rc_timestamp);
1080 CREATE INDEX /*i*/rc_ip ON /*_*/recentchanges (rc_ip);
```

```
1081 CREATE INDEX /*i*/rc_ns_usertext ON /*_*/recentchanges (rc_namespace, rc_user_text);
1082 CREATE INDEX /*i*/rc_user_text ON /*_*/recentchanges (rc_user_text, rc_timestamp);
1083
1084
1085 CREATE TABLE /*_*/watchlist (
1086   -- Key to user.user_id
1087   wl_user int unsigned NOT NULL,
1088
1089   -- Key to page_namespace/page_title
1090   -- Note that users may watch pages which do not exist yet,
1091   -- or existed in the past but have been deleted.
1092   wl_namespace int NOT NULL default 0,
1093   wl_title varchar(255) binary NOT NULL default '',
1094
1095   -- Timestamp when user was last sent a notification e-mail;
1096   -- cleared when the user visits the page.
1097   wl_notificationtimestamp varbinary(14)
1098
1099 ) /*$wgDBTableOptions*/;
1100
1101 CREATE UNIQUE INDEX /*i*/wl_user ON /*_*/watchlist (wl_user, wl_namespace, wl_title);
1102 CREATE INDEX /*i*/namespace_title ON /*_*/watchlist (wl_namespace, wl_title);
1103
1104
1105 --
1106 -- When using the default MySQL search backend, page titles
1107 -- and text are munged to strip markup, do Unicode case folding,
1108 -- and prepare the result for MySQL's fulltext index.
1109 --
1110 -- This table must be MyISAM; InnoDB does not support the needed
1111 -- fulltext index.
1112 --
1113 CREATE TABLE /*_*/searchindex (
1114   -- Key to page_id
1115   si_page int unsigned NOT NULL,
1116
1117   -- Munged version of title
1118   si_title varchar(255) NOT NULL default '',
1119
1120   -- Munged version of body text
1121   si_text mediumtext NOT NULL
1122 ) ENGINE=MyISAM;
1123
1124 CREATE UNIQUE INDEX /*i*/si_page ON /*_*/searchindex (si_page);
1125 CREATE FULLTEXT INDEX /*i*/si_title ON /*_*/searchindex (si_title);
1126 CREATE FULLTEXT INDEX /*i*/si_text ON /*_*/searchindex (si_text);
1127
1128
1129 --
1130 -- Recognized interwiki link prefixes
1131 --
1132 CREATE TABLE /*_*/interwiki (
1133   -- The interwiki prefix, (e.g. "Meatball", or the language prefix "de")
1134   iw_prefix varchar(32) NOT NULL,
1135
1136   -- The URL of the wiki, with "$1" as a placeholder for an article name.
1137   -- Any spaces in the name will be transformed to underscores before
1138   -- insertion.
1139   iw_url blob NOT NULL,
1140
1141   -- The URL of the file api.php
1142   iw_api blob NOT NULL,
1143
1144   -- The name of the database (for a connection to be established with wfGetLB( 'wikiid' ))
1145   iw_wikiid varchar(64) NOT NULL,
1146
1147   -- A boolean value indicating whether the wiki is in this project
1148   -- (used, for example, to detect redirect loops)
1149   iw_local bool NOT NULL,
1150
1151   -- Boolean value indicating whether interwiki transclusions are allowed.
1152   iw_trans tinyint NOT NULL default 0
1153 ) /*$wgDBTableOptions*/;
1154
1155 CREATE UNIQUE INDEX /*i*/iw_prefix ON /*_*/interwiki (iw_prefix);
1156
1157
1158 --
```

```
1159 -- Used for caching expensive grouped queries
1160 --
1161 CREATE TABLE /*_*/querycache (
1162   -- A key name, generally the base name of of the special page.
1163   qc_type varbinary(32) NOT NULL,
1164
1165   -- Some sort of stored value. Sizes, counts...
1166   qc_value int unsigned NOT NULL default 0,
1167
1168   -- Target namespace+title
1169   qc_namespace int NOT NULL default 0,
1170   qc_title varchar(255) binary NOT NULL default ''
1171 ) /*$wgDBTableOptions*/;
1172
1173 CREATE INDEX /*i*/qc_type ON /*_*/querycache (qc_type,qc_value);
1174
1175
1176 --
1177 -- For a few generic cache operations if not using Memcached
1178 --
1179 CREATE TABLE /*_*/objectcache (
1180   keyname varbinary(255) NOT NULL default '' PRIMARY KEY,
1181   value mediumblob,
1182   exptime datetime
1183 ) /*$wgDBTableOptions*/;
1184 CREATE INDEX /*i*/exptime ON /*_*/objectcache (exptime);
1185
1186
1187 --
1188 -- Cache of interwiki transclusion
1189 --
1190 CREATE TABLE /*_*/transcache (
1191   tc_url varbinary(255) NOT NULL,
1192   tc_contents text,
1193   tc_time binary(14) NOT NULL
1194 ) /*$wgDBTableOptions*/;
1195
1196 CREATE UNIQUE INDEX /*i*/tc_url_idx ON /*_*/transcache (tc_url);
1197
1198
1199 CREATE TABLE /*_*/logging (
1200   -- Log ID, for referring to this specific log entry, probably for deletion and such.
1201   log_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
1202
1203   -- Symbolic keys for the general log type and the action type
1204   -- within the log. The output format will be controlled by the
1205   -- action field, but only the type controls categorization.
1206   log_type varbinary(32) NOT NULL default '',
1207   log_action varbinary(32) NOT NULL default '',
1208
1209   -- Timestamp. Duh.
1210   log_timestamp binary(14) NOT NULL default '19700101000000',
1211
1212   -- The user who performed this action; key to user_id
1213   log_user int unsigned NOT NULL default 0,
1214
1215   -- Name of the user who performed this action
1216   log_user_text varchar(255) binary NOT NULL default '',
1217
1218   -- Key to the page affected. Where a user is the target,
1219   -- this will point to the user page.
1220   log_namespace int NOT NULL default 0,
1221   log_title varchar(255) binary NOT NULL default '',
1222   log_page int unsigned NULL,
1223
1224   -- Freeform text. Interpreted as edit history comments.
1225   log_comment varchar(255) NOT NULL default '',
1226
1227   -- LF separated list of miscellaneous parameters
1228   log_params blob NOT NULL,
1229
1230   -- rev_deleted for logs
1231   log_deleted tinyint unsigned NOT NULL default 0
1232 ) /*$wgDBTableOptions*/;
1233
1234 CREATE INDEX /*i*/type_time ON /*_*/logging (log_type, log_timestamp);
1235 CREATE INDEX /*i*/user_time ON /*_*/logging (log_user, log_timestamp);
1236 CREATE INDEX /*i*/page_time ON /*_*/logging (log_namespace, log_title, log_timestamp);
```

```
1237 CREATE INDEX /*i*/times ON /*_*/logging (log_timestamp);
1238 CREATE INDEX /*i*/log_user_type_time ON /*_*/logging (log_user, log_type, log_timestamp);
1239 CREATE INDEX /*i*/log_page_id_time ON /*_*/logging (log_page,log_timestamp);
1240 CREATE INDEX /*i*/type_action ON /*_*/logging(log_type, log_action, log_timestamp);
1241
1242
1243 CREATE TABLE /*_*/log_search (
1244   -- The type of ID (rev ID, log ID, rev timestamp, username)
1245   ls_field varbinary(32) NOT NULL,
1246   -- The value of the ID
1247   ls_value varchar(255) NOT NULL,
1248   -- Key to log_id
1249   ls_log_id int unsigned NOT NULL default 0
1250 ) /*$wgDBTableOptions*/;
1251 CREATE UNIQUE INDEX /*i*/ls_field_val ON /*_*/log_search (ls_field,ls_value,ls_log_id);
1252 CREATE INDEX /*i*/ls_log_id ON /*_*/log_search (ls_log_id);
1253
1254
1255 -- Jobs performed by parallel apache threads or a command-line daemon
1256 CREATE TABLE /*_*/job (
1257   job_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
1258
1259   -- Command name
1260   -- Limited to 60 to prevent key length overflow
1261   job_cmd varbinary(60) NOT NULL default '',
1262
1263   -- Namespace and title to act on
1264   -- Should be 0 and '' if the command does not operate on a title
1265   job_namespace int NOT NULL,
1266   job_title varchar(255) binary NOT NULL,
1267
1268   -- Timestamp of when the job was inserted
1269   -- NULL for jobs added before addition of the timestamp
1270   job_timestamp varbinary(14) NULL default NULL,
1271
1272   -- Any other parameters to the command
1273   -- Stored as a PHP serialized array, or an empty string if there are no parameters
1274   job_params blob NOT NULL
1275 ) /*$wgDBTableOptions*/;
1276
1277 CREATE INDEX /*i*/job_cmd ON /*_*/job (job_cmd, job_namespace, job_title, job_params(128));
1278 CREATE INDEX /*i*/job_timestamp ON /*_*/job(job_timestamp);
1279
1280
1281 -- Details of updates to cached special pages
1282 CREATE TABLE /*_*/querycache_info (
1283   -- Special page name
1284   -- Corresponds to a qc_type value
1285   qci_type varbinary(32) NOT NULL default '',
1286
1287   -- Timestamp of last update
1288   qci_timestamp binary(14) NOT NULL default '19700101000000'
1289 ) /*$wgDBTableOptions*/;
1290
1291 CREATE UNIQUE INDEX /*i*/qci_type ON /*_*/querycache_info (qci_type);
1292
1293
1294 -- For each redirect, this table contains exactly one row defining its target
1295 CREATE TABLE /*_*/redirect (
1296   -- Key to the page_id of the redirect page
1297   rd_from int unsigned NOT NULL default 0 PRIMARY KEY,
1298
1299   -- Key to page_namespace/page_title of the target page.
1300   -- The target page may or may not exist, and due to renames
1301   -- and deletions may refer to different page records as time
1302   -- goes by.
1303   rd_namespace int NOT NULL default 0,
1304   rd_title varchar(255) binary NOT NULL default '',
1305   rd_interwiki varchar(32) default NULL,
1306   rd_fragment varchar(255) binary default NULL
1307 ) /*$wgDBTableOptions*/;
1308
1309 CREATE INDEX /*i*/rd_ns_title ON /*_*/redirect (rd_namespace,rd_title,rd_from);
1310
1311
1312 -- Used for caching expensive grouped queries that need two links (for example double-redirects)
1313 CREATE TABLE /*_*/querycachetwo (
1314   -- A key name, generally the base name of of the special page.
```

```
  qcc_type varbinary(32) NOT NULL,

  -- Some sort of stored value. Sizes, counts...
  qcc_value int unsigned NOT NULL default 0,

  -- Target namespace+title
  qcc_namespace int NOT NULL default 0,
  qcc_title varchar(255) binary NOT NULL default '',

  -- Target namespace+title2
  qcc_namespacetwo int NOT NULL default 0,
  qcc_titletwo varchar(255) binary NOT NULL default ''
) /*$wgDBTableOptions*/;

CREATE INDEX /*i*/qcc_type ON /*_*/querycachetwo (qcc_type,qcc_value);
CREATE INDEX /*i*/qcc_title ON /*_*/querycachetwo (qcc_type,qcc_namespace,qcc_title);
CREATE INDEX /*i*/qcc_titletwo ON /*_*/querycachetwo (qcc_type,qcc_namespacetwo,qcc_titletwo);


-- Used for storing page restrictions (i.e. protection levels)
CREATE TABLE /*_*/page_restrictions (
  -- Page to apply restrictions to (Foreign Key to page).
  pr_page int NOT NULL,
  -- The protection type (edit, move, etc)
  pr_type varbinary(60) NOT NULL,
  -- The protection level (Sysop, autoconfirmed, etc)
  pr_level varbinary(60) NOT NULL,
  -- Whether or not to cascade the protection down to pages transcluded.
  pr_cascade tinyint NOT NULL,
  -- Field for future support of per-user restriction.
  pr_user int NULL,
  -- Field for time-limited protection.
  pr_expiry varbinary(14) NULL,
  -- Field for an ID for this restrictions row (sort-key for Special:ProtectedPages)
  pr_id int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/pr_pagetype ON /*_*/page_restrictions (pr_page,pr_type);
CREATE INDEX /*i*/pr_typelevel ON /*_*/page_restrictions (pr_type,pr_level);
CREATE INDEX /*i*/pr_level ON /*_*/page_restrictions (pr_level);
CREATE INDEX /*i*/pr_cascade ON /*_*/page_restrictions (pr_cascade);


-- Protected titles - nonexistent pages that have been protected
CREATE TABLE /*_*/protected_titles (
  pt_namespace int NOT NULL,
  pt_title varchar(255) binary NOT NULL,
  pt_user int unsigned NOT NULL,
  pt_reason tinyblob,
  pt_timestamp binary(14) NOT NULL,
  pt_expiry varbinary(14) NOT NULL default '',
  pt_create_perm varbinary(60) NOT NULL
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/pt_namespace_title ON /*_*/protected_titles (pt_namespace,pt_title);
CREATE INDEX /*i*/pt_timestamp ON /*_*/protected_titles (pt_timestamp);


-- Name/value pairs indexed by page_id
CREATE TABLE /*_*/page_props (
  pp_page int NOT NULL,
  pp_propname varbinary(60) NOT NULL,
  pp_value blob NOT NULL
) /*$wgDBTableOptions*/;

CREATE UNIQUE INDEX /*i*/pp_page_propname ON /*_*/page_props (pp_page,pp_propname);


-- A table to log updates, one text key row per update.
CREATE TABLE /*_*/updatelog (
  ul_key varchar(255) NOT NULL PRIMARY KEY,
  ul_value blob
) /*$wgDBTableOptions*/;


-- A table to track tags for revisions, logs and recent changes.
CREATE TABLE /*_*/change_tag (
  -- RCID for the change
```

```
1393   ct_rc_id int NULL,
1394   -- LOGID for the change
1395   ct_log_id int NULL,
1396   -- REVID for the change
1397   ct_rev_id int NULL,
1398   -- Tag applied
1399   ct_tag varchar(255) NOT NULL,
1400   -- Parameters for the tag, presently unused
1401   ct_params blob NULL
1402 ) /*$wgDBTableOptions*/;
1403
1404 CREATE UNIQUE INDEX /*i*/change_tag_rc_tag ON /*_*/change_tag (ct_rc_id,ct_tag);
1405 CREATE UNIQUE INDEX /*i*/change_tag_log_tag ON /*_*/change_tag (ct_log_id,ct_tag);
1406 CREATE UNIQUE INDEX /*i*/change_tag_rev_tag ON /*_*/change_tag (ct_rev_id,ct_tag);
1407 -- Covering index, so we can pull all the info only out of the index.
1408 CREATE INDEX /*i*/change_tag_tag_id ON /*_*/change_tag (ct_tag,ct_rc_id,ct_rev_id,ct_log_id);
1409
1410
1411 -- Rollup table to pull a LIST of tags simply without ugly GROUP_CONCAT
1412 -- that only works on MySQL 4.1+
1413 CREATE TABLE /*_*/tag_summary (
1414   -- RCID for the change
1415   ts_rc_id int NULL,
1416   -- LOGID for the change
1417   ts_log_id int NULL,
1418   -- REVID for the change
1419   ts_rev_id int NULL,
1420   -- Comma-separated list of tags
1421   ts_tags blob NOT NULL
1422 ) /*$wgDBTableOptions*/;
1423
1424 CREATE UNIQUE INDEX /*i*/tag_summary_rc_id ON /*_*/tag_summary (ts_rc_id);
1425 CREATE UNIQUE INDEX /*i*/tag_summary_log_id ON /*_*/tag_summary (ts_log_id);
1426 CREATE UNIQUE INDEX /*i*/tag_summary_rev_id ON /*_*/tag_summary (ts_rev_id);
1427
1428
1429 CREATE TABLE /*_*/valid_tag (
1430   vt_tag varchar(255) NOT NULL PRIMARY KEY
1431 ) /*$wgDBTableOptions*/;
1432
1433 -- Table for storing localisation data
1434 CREATE TABLE /*_*/l10n_cache (
1435   -- Language code
1436   lc_lang varbinary(32) NOT NULL,
1437   -- Cache key
1438   lc_key varchar(255) NOT NULL,
1439   -- Value
1440   lc_value mediumblob NOT NULL
1441 ) /*$wgDBTableOptions*/;
1442 CREATE INDEX /*i*/lc_lang_key ON /*_*/l10n_cache (lc_lang, lc_key);
1443
1444 -- Table for caching JSON message blobs for the resource loader
1445 CREATE TABLE /*_*/msg_resource (
1446   -- Resource name
1447   mr_resource varbinary(255) NOT NULL,
1448   -- Language code
1449   mr_lang varbinary(32) NOT NULL,
1450   -- JSON blob
1451   mr_blob mediumblob NOT NULL,
1452   -- Timestamp of last update
1453   mr_timestamp binary(14) NOT NULL
1454 ) /*$wgDBTableOptions*/;
1455 CREATE UNIQUE INDEX /*i*/mr_resource_lang ON /*_*/msg_resource (mr_resource, mr_lang);
1456
1457 -- Table for administering which message is contained in which resource
1458 CREATE TABLE /*_*/msg_resource_links (
1459   mrl_resource varbinary(255) NOT NULL,
1460   -- Message key
1461   mrl_message varbinary(255) NOT NULL
1462 ) /*$wgDBTableOptions*/;
1463 CREATE UNIQUE INDEX /*i*/mrl_message_resource ON /*_*/msg_resource_links (mrl_message, mrl_resource);
1464
1465 -- Table caching which local files a module depends on that aren't
1466 -- registered directly, used for fast retrieval of file dependency.
1467 -- Currently only used for tracking images that CSS depends on
1468 CREATE TABLE /*_*/module_deps (
1469   -- Module name
1470   md_module varbinary(255) NOT NULL,
```

```
1471   -- Skin name
1472   md_skin varbinary(32) NOT NULL,
1473   -- JSON blob with file dependencies
1474   md_deps mediumblob NOT NULL
1475 ) /*$wgDBTableOptions*/;
1476 CREATE UNIQUE INDEX /*i*/md_module_skin ON /*_*/module_deps (md_module, md_skin);
1477
1478 -- Table for holding configuration changes
1479 CREATE TABLE /*_*/config (
1480   -- Config var name
1481   cf_name varbinary(255) NOT NULL PRIMARY KEY,
1482   -- Config var value
1483   cf_value blob NOT NULL
1484 ) /*$wgDBTableOptions*/;
1485 -- Should cover *most* configuration - strings, ints, bools, etc.
1486 CREATE INDEX /*i*/cf_name_value ON /*_*/config (cf_name,cf_value(255));
1487
1488 -- vim: sw=2 sts=2 et
```

Properties

Name	Value
svn:eol-style	native
svn:keywords	Author Date Id Revision
svn:mergeinfo	/branches/JSTesting/maintenance/tables.sql:100352-107913 /branches/REL1_15/phase3/maintenance/tables.sql:51646 /branches/iwtransclusion/phase3/maintenance/tables.sql:68448,69480 /branches/new-installer/phase3/maintenance/tables.sql:43664-66004 /branches/resourceloader/phase3/maintenance/tables.sql:68366-69676,69678-70682,70684-71999,72001-72255,72257-72305,72307-72342 /branches/sqlite/maintenance/tables.sql:58211-58321