Bugzilla Schema for Version 3.4.2

Quick links to table definitions:

1. Introduction

This document describes the Bugzilla database schema for Bugzilla version 3.4.2.

This document is generated automatically by a Python script which constructs and colors the schema tables from the stored results of MySQL queries. For more information about the scripts, see `code.html`.

The purpose of this document is to act as a reference for developers of Bugzilla and of code which interacts with Bugzilla (e.g. P4DTI).

The intended readership is P4DTI developers and Bugzilla developers and administrators.

This document is not confidential.

Please send any comments to <p4dti-comments@ravenbrook.com>, and problem reports to <p4dti-support@ravenbrook.com>.

2. Bugzilla overview

Bugzilla is a defect tracking system, written in Perl with a CGI web GUI. By default it uses MySQL to store its tables. PostgreSQL is also supported.

Bugs

Each defect is called a **bug** and corresponds to one row in the bugs table. It is identified by its number, bugs.bug_id.

Products and components

The work managed by Bugzilla is divided into products. The work for each product is in turn divided into the components of that product. Several properties of a new bug (e.g. ownership) are determined by the product and component to which it belongs. Each component is represented by a row in the components table. Each product is represented by a row in the products table.

Products are grouped by "classification". This is optional and controlled by the parameter 'useclassification'. The classifications are used to help in finding bugs and in constructing meaningful time series, but have no other semantics in Bugzilla. There is a default classification, with ID 1, meaning "Unclassified".

Workflow

Each bug has a status (bugs.bug_status). If a bug has a status which shows it has been resolved, it also has a resolution (bugs.resolution), otherwise the resolution field is empty.

Workflow is configurable. The possible status values are stored in the bug_status table; the transitions in the status_workflow table.

This table shows the possible values and valid transitions of the status field in the default workflow.

Status	Resolved?	Description	Transitions
UNCONFIRMED	No	A new bug, when a product has voting	to NEW by voting or confirmation to ASSIGNED by acceptance to RESOLVED by resolution

NEW	No	Recently added or confirmed	to ASSIGNED by acceptance to RESOLVED by analysis and maybe fixing to NEW by reassignment
ASSIGNED	No	Has been assigned	to NEW by reassignment to RESOLVED by analysis and maybe fixing
REOPENED	No	Was once resolved but has been reopened	to NEW by reassignment to ASSIGNED by acceptance to RESOLVED by analysis and maybe fixing
RESOLVED	Yes	Has been resolved (e.g. fixed, deemed unfixable, etc. See "resolution" column)	to REOPENED by reopening to VERIFIED by verification to CLOSED by closing
VERIFIED	Yes	The resolution has been approved by QA	to CLOSED when the product ships to REOPENED by reopening
CLOSED	Yes	Over and done with	to REOPENED by reopening

This table shows the allowable values of the resolution field. The values "FIXED", "MOVED", and "DUPLICATE" have special meaning for Bugzilla. The other values may be changed, by using editvalues.cgi, to add, remove, or rename values as necessary.

Resolution	**Meaning**
FIXED	The bug has been fixed.
INVALID	The problem described is not a bug.
WONTFIX	This bug will never be fixed.
LATER	This bug will not be fixed in this version.
REMIND	This bug probably won't be fixed in this version.
DUPLICATE	This is a duplicate of an existing bug A description comment is added to this effect, and a record is added to the duplicates table.

WORKSFORME	This bug could not be reproduced.
MOVED	This bug has been moved to another database.

Users

Bugzilla has users. Each user is represented by one row in the profiles table. Each user is referred by a number (profiles.userid) and an email address (profiles.login_name).

Authentication

There are various authentication mechanisms, including "environment variable authentication" (Bugzilla/Auth/Login/WWW/Env.pm) which uses environment variables to pass an external user ID (profiles.extern_id) to the Bugzilla CGI. The rest of this section describes the password-based authentication which has always been in Bugzilla and which is still widely used.

Each user has a password, used to authenticate that user to Bugzilla. The password is stored in profiles.cryptpassword in encrypted form.

On a successful login, Bugzilla generates a pair of cookies for the user's browser. On subsequent accesses, a user gets access if these cookie checks pass:

- they have both Bugzilla_login and Bugzilla_logincookie cookies;
- Their Bugzilla_login is the profiles.userid of a row in the profiles table;
- their Bugzilla_logincookie matches a row in the logincookies table;
- the userids of these two rows match;
- logincookies.ipaddr matches the CGI REMOTE_ADDR;
- profiles.disabledtext is empty.

If the cookie checks fail, the user has to login (with their password), in which case a new row is added to the logincookies table and the user gets a new pair of cookies.

Rows in the logincookies table are deleted after 30 days (at user login time).

Voting

Users may vote for bugs which they think are important. A user can vote for a bug more than once. Votes are recorded in the votes table.

The maximum number of votes per bug per user is product-dependent. Whether or not project managers pay any attention to votes is up to them, apart from the "confirmation by acclamation" process, which is as follows:

New bugs have the status UNCONFIRMED. To enter the main workflow, they need the status NEW. To get the status NEW, they need a particular number of votes which is product-dependent.

Milestones

Products may have "milestones" defined. The intention is that a milestone should be a point in a project at which a set of bugs has been resolved. An example might be a product release or a QA target. Milestones may be turned on and off with the parameter "usetargetmilestone".

If milestones are on, each bug has a "target milestone" (by which it should be fixed). A product may have a URL associated with it which locates a document describing the milestones for that product. This document itself is entirely outside Bugzilla. A product may also have a default target milestone, which is given to new bugs.

Milestones for a product have a "sort key", which allows them to be presented in a specific order in the user interface.

Milestones are kept in the milestones table.

Versions

Products may have versions. This allows more accurate bug reporting: "we saw it in 1.3.7b3".Versions are totally independent of milestones.

Parameters

The operation of Bugzilla is controlled by parameters. These are set in editparams.cgi. The current values are stored in data/params. They are **not** stored in the database.

The set of parameters is defined in the modules in Bugzilla/Config/.

Groups

Bugzilla has "groups" of users. Membership of a group allows a user to perform certain tasks. Each group is represented by a row of the groups table.

There are a number of built-in groups, as follows:

Name	Description

admin	Can administer all aspects of Bugzilla
tweakparams	Can tweak operating parameters
editusers	Can edit or disable users
creategroups	Can create and destroy groups
editcomponents	Can create, destroy, and edit components and other controls (e.g. flagtypes).
editkeywords	Can create, destroy, and edit keywords
editbugs	Can edit all aspects of any bug
canconfirm	Can confirm a bug
editclassifications	Can edit classifications
bz_canusewhines	Can configure whine reports for self
bz_canusewhineatothers	Can configure whine reports for other users
bz_sudoers	Can impersonate another user
bz_sudo_protect	Cannot be impersonated

New groups may be added and used to control access to sets of bugs. These "bug groups" have groups.isbuggroup set to 1. A bug may be in any number of bug groups. To see a bug, a user must be a member of all the bug groups which the bug is in.

If the parameter "usebuggroups" is on, each product automatically has a bug group associated with it. If the parameter "usebuggroupsentry" is also on, a user must be in the product's bug group in order to create new bugs for the product.

Users may be added to a group by any user who has the "bless" property for that group. The "bless" property itself may only be conferred by an administrator.

Group membership for new users and new groups is determined by matching groups.userregexp against the user's email address.The default configuration has universal regexps for the "editbugs" and "canconfirm" groups.

User membership in a group is conferred by a row in the user_group_map table, with user_group_map.isbless set to 0. The bless privilege for a group is conferred by a row with user_group_map.isbless set to 1. Bug membership in a bug group is conferred by a row in the bug_group_map table.

Groups may be configured so that membership in one group automatically confers membership or the "bless" privilege for another group. This is controlled by the group_group_map table.

Groups may be configured so that the existence of a group is not visible to members of another group. This is controlled by the group_group_map table.

A product may be configured so that membership in one or more groups is required to perform certain actions on bugs in the product. Whether or not a new bug for the product is placed in a group is also configurable (note that user membership in a group is required to place an existing bug in that group). All this is controlled by the group_control_map table.

The group_control_map.membercontrol and group_control_map.othercontrol columns of that table determine the treatment of a given group for a new bug in a given product, depending on whether the bug is being created by a member or non-member of that group respectively. The possible values of these columns are as follows:

value	name	meaning
0	NA	A bug for this product cannot be placed in this group.
1	Shown	A bug for this product may be placed in this group, but will not be by default.
2	Default	A bug for this product may be placed in this group, and is by default.
3	Mandatory	A bug for this product is always placed in this group.

Only certain combinations of membercontrol/othercontrol are permitted, as follows:

membercontrol	othercontrol	Notes
0(NA)	0(NA)	A bug for this product can never be placed in this group (so the option isn't presented).
1 (Shown)	0(NA)	Only members can place a bug in this group.**This is the default setting.**
	1 (Shown)	Anyone can place a new bug in this group.
	2 (Default)	Anyone can place a bug in this group, and non-members will do so by default.
	3 (Mandatory)	Anyone can place a bug in this group, and non-members will always do so.
	0(NA)	Only members can place a bug in this group, and do so by default.

2 (Default)	2 (Default)	Anyone can place a bug in this group, and does so by default.
	3 (Mandatory)	Members can place a bug in this group, and do so by default. Non-members always place a bug in this group.
3(Mandatory)	3(Mandatory)	A bug for this product can never be removed from this group (so the option isn't presented).

Attachments

Users can upload attachments to bugs. An attachments can be marked as a patch. Attachments are stored in the attachments table.Attachment data is stored in the attach_data table.Attachments can be URLs, marked by the flag attachments.isurl. The URL itself is stored in attach_data.thedata.

Attachment statuses are implemented with the flags system.

Flags

Bugs and attachments may be marked with "flags". The set of flag types is user-defined (using editflagtypes.cgi). For instance, a flag type might be "candidate for version 7.3 triage", or "7.3" for short. Flag types are recorded in the flagtypes table. Each flag type is either for bugs or for attachments, not both.

Actual flags are recorded in the flags table. Each flag has a status of "+" ("granted"), "-" ("denied") or "?" ("requested"). For instance, one bug might have flag "7.3+", and another might have flag "7.3-".

A status of "?" indicates that a user has requested that this item be given this flag. There is an special interface for viewing request flags (request.cgi). A request flag may be marked for the attention of a particular user, the "requestee".

A flag type may have a "CC list" of email addresses, of people to notify when a flag is requested.

By default, a single bug or attachment may receive several flags of the same type, with the same or different statuses and the same or different requestees. This may be disabled for any given flag type.

Particular flag types may only be available to bugs in certain products and components (or their attachments). This is recorded in the flaginclusions table. Particular flag types may *not* be available to bugs in certain products and components (or their attachments). This is recorded in the flagexclusions table.

Various features of flag types may be disabled: they can be made inactive, not requestable, not "requesteeable", not "multiplicable".

Keywords

Bugzilla users can define a number of keywords, and then give each bug a set of keywords. This is mainly for use in finding related bugs. The keywords are stored in the keyworddefs table, and the one-to-many mapping from bugs to keywords is stored in the keywords table, and also in bugs.keywords.

Dependencies

Bugs may depend on other bugs being fixed. That is, it may be impossible to fix one bug until another one is fixed. Bugzilla records and displays such information and uses it to notify users when a bug changes (all contacts for all dependent bugs are notified when a bug changes).

Dependencies are recorded in the dependencies table.

Activity

Bugzilla keeps a record of changes made to bugs. This record is in the bugs_activity table. Each row in this table records a change to a field in the bugs table.The fields are referred to by a number which is looked up in the fielddefs table. This table records the name of the field and also a longer description used to display activity tables.

Severity

Each bug has a "severity" field, bugs.bug_severity, indicating the severity of the impact of the bug. There is no code in Bugzilla which distinguishes the values of this field, although it may naturally be used in queries.The set of values available for this field is stored in bug_severity and can be controlled by the administrator. The intended meanings of the built-in values of this field are as follows:

Value	**Intended meaning**
Blocker	Blocks development and/or testing work
Critical	Crashes, loss of data, severe memory leak
Major	Major loss of function
Minor	Minor loss of function, or other problem where easy workaround is present
Trivial	Cosmetic problem
Enhancement	Request for enhancement

Email notification

When a bug changes, email notification is sent out to a number of users:

- The bug's owner (bugs.assigned_to)
- The bug's reporter (bugs.reporter)
- The bug's QA contact, if the "useqacontact" parameter is set (bugs.qa_contact)
- All the users who have explicitly asked to be notified when the bug changes (these users are stored in the cc table).
- All the users who have voted for this bug (recorded in the votes table).

Individual users may filter these messages according to the way in which the bug changes and their relationship to the bug. These filtering preferences are recorded in the email_setting table.

This is handled by the Bugzilla::Bugmail module, which is invoked by the template system (from Bugzilla::Template) when it encounters a call to SendBugMail() in a template.

If the parameter "use_mailer_queue" is set, all email is queued to be sent asynchronously. This is managed by a third-party general-purpose Perl job queueing system called TheSchwartz, using several database tables of its own (ts_error, ts_exitstatus, ts_funcmap, ts_job, and ts_note).

Long descriptions

Each bug has a number of comments associated with it. These are stored individually in the longdescs table.

They are displayed as the "Description" on the bug form, ordered by date and annotated with the user and date. Users can add new comments with the "Additional comment" field on the bug form.

Named queries

Users can name queries. Links to named query pages appear in a navigation footer bar on most Bugzilla pages. A query named "(Default query)" is a user's default query. Named queries are stored in the namedqueries table.

If the parameter "querysharegroup" is set, it names a group of users who are empowered to share named queries. An empowered user can share a given named query they create with all the members of a group, as long as he or she has the "bless" property for that group. A query can only be shared with a single group. Sharing is recorded in the namedquery_group_map table.

Any user able to use a given named query can control whether or not that query appears in his or her navigation footer bar. This is recorded

in the namedqueries_link_in_footer table.

Charts

Bugzilla can draw general time-series charts. There are a number of default time series. Each product has a default series for each bug status or resolution (for instance "how many bugs are INVALID in product Foo") and each component has a default series for all open bugs (UNCONFIRMED/NEW/ASSIGNED/REOPENED) and one for all closed bugs (RESOLVED/VERIFIED/CLOSED). A user can also define a new time series based on any query, and give it a "frequency" (actually a period, measured in days). The set of series is stored in the series table.

To collect the data for the time series, the Bugzilla administrator needs to arrange for the collectstats.pl script to be run every day. This script stores the data in the series_data table.

Series have categories and subcategories, which are provided in order to make it easier to manage large numbers of series. They are normalized in the series_categories table.

By default, a time series is "private": only visible to the user who created it. An administrator may make a time series "public", or visible to other users.this is determined by series.public.

Visibility of a time series to a user is determined on a per-category basis using the groups system. The group memberships required to see a time series in a given category are recorded in the category_group_map table. A user may see a time series if they are in all the groups for the category *and* either ths user created the series or it is public.

Watchers

Bugzilla lets users "watch" each other; receiving each other's Bugzilla email. For instance, if Sam goes on holiday, Phil can "watch" her, receiving all her Bugzilla email. This is set up by the user preferences (userprefs.cgi), recorded in the watch table and handled by the email subsystem.

Time tracking

Bugzilla can track time for each bug, if the "timetrackinggroup" parameter is set. Members of that group get the ability to estimate the amount of effort (measured in hours) a bug will take to fix, either when creating or when editing the bug. Members of that group are also permitted to record hours of effort spent on the bug

longdescs.work_time records each increment of work. The sum of this column for a bug is computed to display as "Hours Worked" for the bug.

bugs.estimated_time is the estimate for how much time the bug will take in total, displayed as "Orig. Est.". This can be changed by members of the timetrackinggroup.

bugs.remaining_time is the current estimate for how much more time the bug will take to fix, displayed as "Hours Left". This can be changed by members of the timetrackinggroup.

The total of "Hours Left" and "Hours Worked" is shown as "Current Est.": the current estimate of the total effort required to fix the bug. "Hours Worked" as a percentage of "Current Est" is shown as "% Complete". "Current Est" deducted from "Orig. Est" is shown as "Gain"

bugs.deadline records a calendar deadline for the bug.

The Whine System

Bugzilla has a system for sending "whine" email messages to specified users on a regular basis. This system relies on the administrator configuring the Bugzilla server to run a script at regular intervals (e.g. by using crontab).

The `whineatnews.pl` script should be run once a day. For each bug which has status NEW or REOPENED, and which has not changed for a certain number of days, it sends a message to the bug's owner. The number of days is controlled by a Bugzilla parameter called "whinedays". The content of the email message is controlled by a Bugzilla parameter called "whinemail".

The `whine.pl` script runs a separate whine system, which allows a number of whine schedules to be established with varying frequency (up to every 15 minutes), criteria, and content of whine messages. It is configured with `editwhines.cgi`. Obviously, `whine.pl` needs to be run every 15 minutes in order to send the most frequent messages.

Users must be in the bz_canusewhines group to configure whine messages. Users must be in the bz_canusewhineatothers group to configure whine messages *to be sent to other users*. These restrictions are checked when configuring whine messages and also before messages are sent, so removing a user from one of these groups will disable any whines which that user has configured.

A whine schedule, stored in the whine_schedules table, specifies the frequency with which an email should be sent to a particular user. The email is specified with a whine event (see below). There is a variety of ways of specifying the frequency: both days (every day, a particular day of the week, weekdays only, a particular day of the month, the last day of the month) and times (a particular hour, or every 15, 30, or 60 minutes).

Whines may be scheduled for groups as well as users.

A whine schedule, stored in the whine_schedules table, specifies the frequency with which an email should be sent to a particular user. The email is specified with a whine event (see below). There is a variety of ways of specifying the frequency: both days (every day, a particular day of the week, weekdays only, a particular day of the month, the last day of the month) and times (a particular hour, or every 15, 30, or 60 minutes).

A whine event, stored in the whine_events table, describes an email message: subject line and some body text to precede query results. A message may consist of more than one whine query.

A whine query, stored in the whine_queries table is a named query, to which a title is given for use in email messages. Whine queries are stored in the whine_queries table. A whine query may specify that a separate message is to be sent for each bug found.

Settings

There are several user-interface preferences, each of which can take a number of values. Each preference has a row in the setting table, and possible values in the setting_value table. The administrator may set a default value for each preference (setting.default_value) and determine whether users are able to override the default (setting.is_enabled). The user's individual preferences are recorded in the profile_setting table.

Quips

Bugzilla supports "quips": small text messages, often humorous, which appear along with search results. The quips are selected at random from a set.

The quips are stored in the quips table.

Quips may be entered or deleted using `quips.cgi`.

Quips may be entered by any user but must be approved by an administrator before they can be displayed.

References to other Bugzillas

Bugzilla can record connections to bugs in other instances of Bugzilla, if the parameter "use_see_also" is set. The connections are displayed as clickable URLs and are stored as URLs in the bug_see_also table. They are validated according to the system's notion of a valid form for Bugzilla URLs.

Custom Fields

Bugzilla supports custom fields. Each custom field is a new column in the bugs table, with a name beginning `cf_`. The presence of each custom field is indicated by a row in the fielddefs table, with fielddefs.custom set to 1. The type of each custom field is specified by fielddefs.type:

The value 1 (FIELD_TYPE_FREETEXT) indicates a free-form text field (type varchar(255)).

The value 2 (FIELD_TYPE_SINGLE_SELECT) indicates a single-select field (type varchar(64), not null, default '---'). The allowable values of that field are stored in a special table with the same `cf_<name>` name as the field, and a schema like this:

Field	Type	Default	Properties	Remarks
id	smallint	0	auto_increment	a unique ID.
value	varchar(64)	''	-	the text value
sortkey	smallint	0	-	a number determining the order in which values appear.
isactive	tinyint	1	-	1 if this value is currently available, 0 otherwise
visibility_value_id	smallint	0	-	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
cf_<field>_value_idx	value	unique	-
cf_<field>_sortkey_idx	sortkey	-	-
cf_<field>_visibility_value_id_idx	visibility_value_id	-	-

The value 3 (FIELD_TYPE_MULTI_SELECT) indicates a multi-select field. The allowable values of that field are stored in a `cf_<name>` table as for FIELD_TYPE_SINGLE_SELECT, above. The actual values of the field are not stored in the bugs table, unlike other custom fields, Instead they are stored in another table, with the name `bug_cf_<name>`, and a schema like this:

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0		The bug ID (foreign key bugs.bug_id).
value	varchar(64)	''	-	the value (foreign key cf_<name>.value).

Indexes:

Name	Fields	Properties	Remarks
cf_<field>_bug_id_idx	bug_id, value	unique	-

The value 4 (FIELD_TYPE_TEXTAREA) indicates a large text-box field (type mediumtext).

The value 5 (FIELD_TYPE_DATETIME) indicates a date/time field (type datetime).

The value 6 (FIELD_TYPE_BUG_ID) indicates a bug ID field (type mediumint).

Custom fields are configured using `editfield.cgi`.

List of tables

Name	Description
attach_data	The content of attachments.
attachments	Bug attachments.
bug_group_map	Which bugs are in which groups. See the notes on groups.
bug_see_also	Related bugs in other Bugzillas.
bug_severity	The severity values of bugs.
bug_status	The status values of bugs.
bugs	The bugs themselves.
bugs_activity	Activity on the bugs table.
bugs_fulltext	All the descriptive text

Name	Description
namedqueries	Named queries.
namedqueries_link_in_footer	Controls whether a named query appears in a given user's navigation footer.
namedquery_group_map	Controls whether a named query is shared with other users (other members of a group).
op_sys	The possible values of the "operating system" field of a bug.
priority	The possible values of the "priority" field of a bug.
products	One row for each product. See the notes on products.
profile_setting	User preference settings.
profiles	Describes Bugzilla users. One row per user.
profiles_activity	This table is for recording changes to the profiles table. Currently it only records changes to group membership made with

	on bugs, to speed up searching.
bz_schema	The database schema itself.
category_group_map	Which groups does a user have to be in to view chart data in a given category. See the notes on charts.
cc	Users who have asked to receive email when a bug changes.
classifications	Product classifications. See the notes on products.
component_cc	Users to put on the CC list for a new bug in a given component.
components	One row for each component. See the notes on products and components.
dependencies	Which bugs depend on other bugs.
duplicates	Which bugs are duplicates of which other bugs.
email_setting	Per-user settings controlling when email is sent to that user.
fielddefs	The properties of each

	editusers.cgi. This allows the administrator to track group inflation. There is currently no code to inspect this table; only to add to it.
quips	A table of quips.
rep_platform	The possible values of the "platform" field of a bug.
resolution	The possible values of the "resolution" field of a bug.
series	Properties of the time-series datasets available (e.g. for plotting charts). See the notes on charts.
series_categories	
series_data	Data for plotting time-series charts. See the notes on charts.
setting	Identifies the set of user preferences.
setting_value	Possible values for user preferences.
status_workflow	Identifies allowable workflow transitions.
tokens	Tokens are sent to users to track activities such as creating new accounts and changing email addresses or passwords. They are also sent to browsers and used to track workflow, to prevent security problems (e.g. so that one can only delete groups from a session last seen on a group management page).
ts_error	A log of errors from TheSchwartz asynchronous job-queueing system. Rows are aged out of this table after seven days.
ts_exitstatus	A log of job completions from TheSchwartz asynchronous job-queueing system.

	bug field.
flagexclusions	It may be forbidden to set a given flag on an item (bug or attachment) if that item is in a given product and/or component. This table records such exclusions. See the notes on flags.
flaginclusions	An item (bug or attachment) may be required to be in a given product and/or component for a flag to be set. This table records such requirements. See the notes on flags.
flags	This table records the flags set on bugs or attachments. See the notes on flags.
flagtypes	The types of flags available for bugs and attachments. See the notes on flags.
group_control_map	This table describes the relationship of groups to products (whether membership in a given group is required for entering or editing a bug in a given product). See the notes on groups.
group_group_map	Groups can be

ts_funcmap	The table of functions for TheSchwartz asynchronous job-queueing system.
ts_job	The job queue managed by TheSchwartz asynchronous job-queueing system.
ts_note	Notes on jobs for TheSchwartz asynchronous job-queueing system. Apparently not used.
user_group_map	This table records which users are members of each group, or can "bless" each group. See the notes on groups.
versions	Product versions.
votes	votes.
watch	watchers.
whine_events	One row for each regular whine event. See the notes on whining.
whine_queries	See the notes on whining.
whine_schedules	See the notes on whining.

	configured such that membership of one group automatically confers rights over some other groups. This table records that configuration. See the notes on groups.
groups	This table describes a number of user groups. Each group allows its members to perform a restricted activity. See the notes on groups.
keyworddefs	Names and definitions of the keywords. See the notes on keywords.
keywords	Bugs may have keywords. This table defines which bugs have which keywords. The keywords are defined in the keyworddefs table.
logincookies	Bugzilla generates a cookie each time a user logs in, and uses it for subsequent authentication. The cookies generated are stored in this table. For more information, see the notes on authentication.
longdescs	Long bug descriptions.
milestones	Development milestones.

3. The schema

The "attach_data" table

The content of attachments.

Field	Type	Default	Properties	Remarks
id	mediumint	0	-	The attachment id (foreign key attachments.attach_id).
thedata	longblob	''	-	the content of the attachment.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-

The "attachments" table

Bug attachments.

Field	Type	Default	Properties	Remarks
attach_id	mediumint	None	auto_increment	a unique ID.
bug_id	mediumint	0	-	the bug to which this is attached (foreign key bugs.bug_id)
creation_ts	datetime	0000-00-00 00:00:00	-	the creation time.
description	tinytext	''	-	a description of the attachment.
filename	varchar(100)	''	-	the filename of the attachment.

isobsolete	tinyint	0	-	Non-zero if this attachment is marked as obsolete.
ispatch	tinyint	None	null	non-zero if this attachment is a patch file.
isprivate	tinyint	0	-	Non-zero if this attachment is "private", i.e. only visible to members of the "insider" group.
isurl	tinyint	0	-	Non-zero if this attachment is actually a URL.
mimetype	tinytext	''	-	the MIME type of the attachment.
modification_time	datetime	0000-00-00 00:00:00	-	the modification time of the attachment.
submitter_id	mediumint	0	-	the userid of the attachment (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	attach_id	unique	-
attachments_modification_time_idx	modification_time		-
attachments_submitter_id_idx	submitter_id, bug_id		-
attachments_bug_id_idx	bug_id		-
attachments_creation_ts_idx	creation_ts		-

The "bug_group_map" table

Which bugs are in which groups. See the notes on groups.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	The bug id, (foreign key bugs.bug_id)
group_id	mediumint	0	-	The group id, (foreign key groups.id)

Indexes:

Name	Fields	Properties	Remarks
bug_group_map_bug_id_idx	bug_id, group_id	unique	-
bug_group_map_group_id_idx	group_id		-

The "bug_see_also" table

Related bugs in other Bugzillas.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	The bug id, (foreign key bugs.bug_id)
value	varchar(255)	''	-	The URL of a related bug in another Bugzilla.

Indexes:

Name	Fields	Properties	Remarks
bug_see_also_bug_id_idx	bug_id, value	unique	-

The "bug_severity" table

The severity values of bugs.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.
isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field
visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id,

				for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
bug_severity_sortkey_idx	sortkey, value		-
bug_severity_value_idx	value	unique	-
bug_severity_visibility_value_id_idx	visibility_value_id		-

The "bug_status" table

The status values of bugs.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.
is_open	tinyint	1	-	1 if the status is "Open", 0 if it is "Closed".
isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field
visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-

bug_status_sortkey_idx	sortkey, value		-
bug_status_value_idx	value	unique	-
bug_status_visibility_value_id_idx	visibility_value_id		-

The "bugs" table

The bugs themselves.

Field	Type	Default	Properties	Remarks
alias	varchar(20)	None	null	An alias for the bug which can be used instead of the bug number.
assigned_to	mediumint	0	-	The current owner of the bug (foreign key profiles.userid).
bug_file_loc	mediumtext	None	null	A URL which points to more information about the bug.
bug_id	mediumint	None	auto_increment	The bug ID.
bug_severity	varchar(64)	''	-	See the notes. foreign key bug_severity.value.
bug_status	varchar(64)	''	-	The workflow status of the bug. foreign key bug_status.value.
cclist_accessible	tinyint	1	-	1 if people on the CC list can see this bug (even if in the wrong group); 0 otherwise.
component_id	smallint	0	-	The product component (foreign key components.id)
creation_ts	datetime	None	null	The times of the bug's creation.
deadline	datetime	None	null	The deadline for this bug (a date).
delta_ts	datetime	0000-00-00 00:00:00	-	The timestamp of the last update. This includes updates to some related tables (e.g. the longdescs table).
estimated_time	decimal(5,2)	0.0	-	The original estimate of the total effort required to fix this bug (in hours).
everconfirmed	tinyint	0	-	1 if this bug has ever been confirmed. This is used for validation of some sort.

keywords	mediumtext	''	-	A set of keywords. Note that this duplicates the information in the keywords table. (foreign key keyworddefs.name)
lastdiffed	datetime	None	null	The time at which information about this bug changing was last emailed to the cc list.
op_sys	varchar(64)	''	-	The operating system on which the bug was observed. foreign key op_sys.value.
priority	varchar(64)	''	-	The priority of the bug. foreign key priority.value.
product_id	smallint	0	-	The product (foreign key products.id)
qa_contact	mediumint	None	null	The QA contact (foreign key profiles.userid)
remaining_time	decimal(5,2)	0.0	-	The current estimate of the remaining effort required to fix this bug (in hours).
rep_platform	varchar(64)	''	-	The platform on which the bug was reported. foreign key rep_platform.value.
reporter	mediumint	0	-	The user who reported this (foreign key profiles.userid)
reporter_accessible	tinyint	1	-	1 if the reporter can see this bug (even if in the wrong group); 0 otherwise.
resolution	varchar(64)	''	-	The bug's resolution foreign key resolution.value.
short_desc	varchar(255)	''	-	A short description of the bug.
status_whiteboard	mediumtext	''	-	This seems to be just a small whiteboard field.
target_milestone	varchar(20)	---	-	The milestone by which this bug should be resolved. (foreign key milestones.value)
version	varchar(64)	''	-	The product version (foreign key versions.value)
votes	mediumint	0	-	The number of votes.

Indexes:

Name	Fields	Properties	Remarks

PRIMARY	bug_id	unique	-
bugs_alias_idx	alias	unique	-
bugs_assigned_to_idx	assigned_to		-
bugs_bug_severity_idx	bug_severity		-
bugs_bug_status_idx	bug_status		-
bugs_component_id_idx	component_id		-
bugs_creation_ts_idx	creation_ts		-
bugs_delta_ts_idx	delta_ts		-
bugs_op_sys_idx	op_sys		-
bugs_priority_idx	priority		-
bugs_product_id_idx	product_id		-
bugs_qa_contact_idx	qa_contact		-
bugs_reporter_idx	reporter		-
bugs_resolution_idx	resolution		-
bugs_target_milestone_idx	target_milestone		-
bugs_version_idx	version		-
bugs_votes_idx	votes		-

The "bugs_activity" table

Activity on the bugs table.

Field	Type	Default	Properties	Remarks
added	tinytext	None	null	The new value of this field, or values which have been added for multi-value fields such as bugs.keywords, the cc table, and the dependencies table

attach_id	mediumint	None	null	If the change was to an attachment, the ID of the attachment (foreign key attachments.attach_id)
bug_id	mediumint	0	-	Which bug (foreign key bugs.bug_id)
bug_when	datetime	0000-00-00 00:00:00	-	When was the change made?
fieldid	mediumint	0	-	What was the fieldid? (foreign key fielddefs.id)
removed	tinytext	None	null	The old value of this field, or values which have been removed for multi-value fields such as bugs.keywords, the cc table, and the dependencies table
who	mediumint	0	-	Which user (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
bugs_activity_bug_id_idx	bug_id		-
bugs_activity_bug_when_idx	bug_when		-
bugs_activity_who_idx	who		-
bugs_activity_fieldid_idx	fieldid		-

The "bugs_fulltext" table

All the descriptive text on bugs, to speed up searching.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	Which bug (foreign key bugs.bug_id)
comments	mediumtext	None	null	The bug's comments, concatenated (longdescs.thetext)
comments_noprivate	mediumtext	None	null	Those comments visible to non-members of the "insider" group (i.e. with longdescs.isprivate zero).
short_desc	varchar(255)	''	-	The bug's short description (bugs.short_desc)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	bug_id	unique	-
bugs_fulltext_comments_idx	comments	full text	-
bugs_fulltext_comments_noprivate_idx	comments_noprivate	full text	-
bugs_fulltext_short_desc_idx	short_desc	full text	-

The "bz_schema" table

The database schema itself.

Field	Type	Default	Properties	Remarks
schema_data	longblob	''	-	A Perl Storable (serialized version) of the abstract schema.
version	decimal(3,2)	0.0	-	The version number of the abstract schema data structures. This is *not* the schema version; it does not change as tables, columns, and indexes are added and removed.

The "bz_schema" table has no indexes.

The "category_group_map" table

Which groups does a user have to be in to view chart data in a given category. See the notes on charts.

Field	Type	Default	Properties	Remarks
category_id	smallint	0	-	The series category (foreign key series_categories.id)
group_id	mediumint	0	-	The group. (foreign key groups.id)

Indexes:

Name	Fields	Properties	Remarks
category_group_map_category_id_idx	category_id, group_id	unique	-

The "cc" table

Users who have asked to receive email when a bug changes.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	The bug (foreign key bugs.bug_id)
who	mediumint	0	-	The user (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
cc_bug_id_idx	bug_id, who	unique	-
cc_who_idx	who		-

The "classifications" table

Product classifications. See the notes on products.

Field	Type	Default	Properties	Remarks
description	mediumtext	None	null	A description of the classification
id	smallint	None	auto_increment	The classification id.
name	varchar(64)	''	-	The classification name.
sortkey	smallint	0	-	A number used to determine the order in which classifications are shown.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
classifications_name_idx	name	unique	-

The "component_cc" table

Users to put on the CC list for a new bug in a given component.

Field	Type	Default	Properties	Remarks
component_id	smallint	0	-	The component id (foreign key components.id).
user_id	mediumint	0	-	The user id (foreign key profiles.userid).

Indexes:

Name	Fields	Properties	Remarks
component_cc_user_id_idx	component_id, user_id	unique	-

The "components" table

One row for each component. See the notes on products and components.

Field	Type	Default	Properties	Remarks
description	mediumtext	''	-	A description of the component.
id	smallint	None	auto_increment	The component id.
initialowner	mediumint	0	-	The default initial owner of bugs in this component. On component creation, this is set to the user who creates the component. foreign key profiles.userid.
initialqacontact	mediumint	None	null	The initial "qa_contact" field for bugs of this component. Note that the use of the qa_contact field is optional, parameterized by Param("useqacontact"). foreign key profiles.userid.

name	varchar(64)	''	-	The component id.
product_id	smallint	0	-	The product (foreign key products.id)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
components_name_idx	name		-
components_product_id_idx	product_id, name	unique	-

The "dependencies" table

Which bugs depend on other bugs.

Field	Type	Default	Properties	Remarks
blocked	mediumint	0	-	Which bug is blocked (foreign key bugs.bug_id)
dependson	mediumint	0	-	Which bug does it depend on (foreign key bugs.bug_id)

Indexes:

Name	Fields	Properties	Remarks
dependencies_blocked_idx	blocked		-
dependencies_dependson_idx	dependson		-

The "duplicates" table

Which bugs are duplicates of which other bugs.

Field	Type	Default	Properties	Remarks

dupe	mediumint	0	-	The duplicate bug (foreign key bugs.bug_id)
dupe_of	mediumint	0	-	The bug which is duplicated (foreign key bugs.bug_id)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	dupe	unique	-

The "email_setting" table

Per-user settings controlling when email is sent to that user.

Field	Type	Default	Properties	Remarks
event	tinyint	0	-	The event on which an email should be sent. 1: added or removed from this capacity; 2: new comments are added; 3: new attachment is added; 4: attachment data is changed; 5: severity, priority, status, or milestone are changed; 6: resolved or reopened; 7: keywords change; 8: CC list changed; 0: any other change. These are overridden and an email is not sent in the following circumstances, unless a suitable row is also present: 50: if the bug is unconfirmed; 51: if the change was by this user. Global events are 100: a flag has been requested of this user; 101: This user has requested a flag.
relationship	tinyint	0	-	The relationship between the user and the bug. 0: Assignee; 1: QA contact; 2: Reporter; 3: CC; 4: Voter; 100: for global events, which do not depend on a relationship.
user_id	mediumint	0	-	The user to whom this setting applies (foreign key profiles.userid).

Indexes:

Name	Fields	Properties	Remarks
email_setting_user_id_idx	user_id, relationship, event	unique	-

The "fielddefs" table

The properties of each bug field.

Field	Type	Default	Properties	Remarks
buglist	tinyint	0	-	1 for a field which can be used as a display or order column in a bug list, 0 otherwise.
custom	tinyint	0	-	1 for a custom field, 0 otherwise. Part of the custom fields system.
description	tinytext	''	-	long description
enter_bug	tinyint	0	-	1 for a field which is present on the bug entry form, 0 otherwise.
id	mediumint	None	auto_increment	primary key for this table
mailhead	tinyint	0	-	whether or not to send the field description in mail notifications.
name	varchar(64)	''	-	field name or definition (some fields are names of other tables or of fields in other tables).
obsolete	tinyint	0	-	1 if this field no longer exists, 0 otherwise.
sortkey	smallint	0	-	the order of fields in mail notifications.
type	smallint	0	-	The field type. 0 (FIELD_TYPE_UNKNOWN) for most non-custom fields. 1 (FIELD_TYPE_FREETEXT) for a single-line text field. 2 (FIELD_TYPE_SINGLE_SELECT) for a single-select field. 3 (FIELD_TYPE_MULTI_SELECT) for a multi-select field. 4 (FIELD_TYPE_TEXTAREA) for a large text box field. 5 (FIELD_TYPE_DATETIME) for a date/time field. 6 (FIELD_TYPE_BUG_ID) for a bug ID field. 7 (FIELD_TYPE_BUG_URLS) for a list of bug URLs.
value_field_id	mediumint	None	null	If not NULL, the ID of a (single-select or multi-select) *chooser field*, which controls the visibility of individual values of this field. Only applies to single-select and multi-select fields. Foreign ney fielddefs.id.
visibility_field_id	mediumint	None	null	If not NULL, the ID of a (single-select or multi-select) *control field* which

				controls the visibility of this field. Only applies to custom fields. Foreign key fielddefs.id.
visibility_value_id	smallint	None	null	If not NULL, and the control field (with ID visibility_field_id) does not have a value with this ID, this field is not visible. Only applies to custom fields. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
fielddefs_value_field_id_idx	value_field_id		-
fielddefs_name_idx	name	unique	-
fielddefs_sortkey_idx	sortkey		-

The "flagexclusions" table

It may be forbidden to set a given flag on an item (bug or attachment) if that item is in a given product and/or component. This table records such exclusions. See the notes on flags.

Field	Type	Default	Properties	Remarks
component_id	smallint	None	null	The component, or NULL for "any". (foreign key components.id)
product_id	smallint	None	null	The product, or NULL for "any". (foreign key products.id)
type_id	smallint	0	-	The flag type. (foreign key flagtypes.id)

Indexes:

Name	Fields	Properties	Remarks
flagexclusions_type_id_idx	type_id, product_id, component_id		-

The "flaginclusions" table

An item (bug or attachment) may be required to be in a given product and/or component for a flag to be set. This table records such requirements. See the notes on flags.

Field	Type	Default	Properties	Remarks
component_id	smallint	None	null	The component, or NULL for "any". (foreign key components.id)
product_id	smallint	None	null	The product, or NULL for "any". (foreign key products.id)
type_id	smallint	0	-	The flag type. (foreign key flagtypes.id)

Indexes:

Name	Fields	Properties	Remarks
flaginclusions_type_id_idx	type_id, product_id, component_id		-

The "flags" table

This table records the flags set on bugs or attachments. See the notes on flags.

Field	Type	Default	Properties	Remarks
attach_id	mediumint	None	null	The attachment, or NULL if this flag is not on an attachment. (foreign key attachments.attach_id)
bug_id	mediumint	0	-	The bug. (foreign key bugs.bug_id)
creation_date	datetime	0000-00-00 00:00:00	-	The date the flag was created.
id	mediumint	None	auto_increment	A unique ID.
modification_date	datetime	None	null	The date the flag was most recently modified or created.
requestee_id	mediumint	None	null	The ID of the user to whom this request flag is addressed, or NULL for non-requestee flags (foreign key profiles.userid)

setter_id	mediumint	None	null	The ID of the user who created, or most recently modified, this flag (foreign key profiles.userid)
status	char(1)	''	-	'+' (granted), '-' (denied), or '?' (requested).
type_id	smallint	0	-	The flag type. (foreign key flagtypes.id)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
flags_bug_id_idx	bug_id, attach_id		-
flags_type_id_idx	type_id		-
flags_requestee_id_idx	requestee_id		-
flags_setter_id_idx	setter_id		-

The "flagtypes" table

The types of flags available for bugs and attachments. See the notes on flags.

Field	Type	Default	Properties	Remarks
cc_list	varchar(200)	None	null	A string containing email addresses to which notification of requests for this flag should be sent. This is filtered using the groups system before messages are actually sent, so that users not entitled to see a bug don't receive notifications concerning it.
description	mediumtext	''	-	The description of the flag
grant_group_id	mediumint	None	null	Group membership required to grant this flag. (foreign key groups.id)
id	smallint	None	auto_increment	The flag type ID
is_active	tinyint	1	-	1 if the flag appears in the UI and can be set; 0 otherwise.
is_multiplicable	tinyint	0	-	1 if multiple instances of this flag may be set on the same item; 0

				otherwise.
is_requestable	tinyint	0	-	1 if the flag may be requested; 0 otherwise.
is_requesteeble	tinyint	0	-	1 if a request for this flag may be aimed at a particular user; 0 otherwise.
name	varchar(50)	''	-	The short flag name
request_group_id	mediumint	None	null	Group membership required to request this flag. (foreign key groups.id)
sortkey	smallint	0	-	An integer used for sorting flags for display.
target_type	char(1)	b	-	'a' for attachment flags, 'b' for bug flags

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-

The "group_control_map" table

This table describes the relationship of groups to products (whether membership in a given group is required for entering or editing a bug in a given product). See the notes on groups.

Field	Type	Default	Properties	Remarks
canconfirm	tinyint	0	-	1 if membership of this group enables confirmation of bugs in this product; 0 otherwise.
canedit	tinyint	0	-	1 if membership of this group is required to edit a bug in this product; 0 otherwise.
editbugs	tinyint	0	-	1 if membership of this group enables editing bugs in this product; 0 otherwise. Note: membership of all 'canedit' groups is also required.
editcomponents	tinyint	0	-	1 if membership of this group enables editing product-specific configuration such as components and flagtypes; 0 otherwise.
entry	tinyint	0	-	1 if membership of this group is required to enter a bug in this product; 0 otherwise.
group_id	mediumint	0	-	The group. (foreign key groups.id)

membercontrol	tinyint	0	-	Determines what control members of this group have over whether a bug for this product is placed in this group. 0 (NA/no control): forbidden. 1 (Shown): permitted. 2 (Default): permitted and by default. 3 (Mandatory): always.
othercontrol	tinyint	0	-	Determines what control non-group-members have over whether a new bug for this product is placed in this group. Group membership of existing bugs can only be changed by members of the relevant group. 0 (NA/no control): forbidden. 1 (Shown): permitted. 2 (Default): permitted and by default. 3 (Mandatory): always. Allowable values depend on the value of membercontrol. See the notes on groups.
product_id	smallint	0	-	The product. (foreign key products.id)

Indexes:

Name	Fields	Properties	Remarks
group_control_map_group_id_idx	group_id		-
group_control_map_product_id_idx	product_id, group_id	unique	-

The "group_group_map" table

Groups can be configured such that membership of one group automatically confers rights over some other groups. This table records that configuration. See the notes on groups.

Field	Type	Default	Properties	Remarks
grant_type	tinyint	0	-	0 if membership is granted; 1 if just "bless" privilege is granted ("bless" does not imply membership), 2 if visibility is granted.
grantor_id	mediumint	0	-	The group whose membership or "bless" privilege is automatically granted.(foreign key groups.id)
member_id	mediumint	0	-	The group whose membership grants membership or "bless" privilege for another group. (foreign key groups.id)

Indexes:

Name	Fields	Properties	Remarks
group_group_map_member_id_idx	member_id, grantor_id, grant_type	unique	-

The "groups" table

This table describes a number of user groups. Each group allows its members to perform a restricted activity. See the notes on groups.

Field	Type	Default	Properties	Remarks
description	mediumtext	''	-	A long description of the group.
icon_url	tinytext	None	null	The URL of an icon for the group (e.g. to be shown next to bug comments made by members of the group).
id	mediumint	None	auto_increment	The group id
isactive	tinyint	1	-	1 if bugs can be added to this group; 0 otherwise.
isbuggroup	tinyint	0	-	1 if this is a group controlling access to a set of bugs.
name	varchar(255)	''	-	A short name for the group.
userregexp	tinytext	''	-	a regexp used to determine membership of new users.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
groups_name_idx	name	unique	-

The "keyworddefs" table

Names and definitions of the keywords. See the notes on keywords.

Field	Type	Default	Properties	Remarks

description	mediumtext	None	null	The meaning of the keyword.
id	smallint	None	auto_increment	A unique number identifying this keyword.
name	varchar(64)	''	-	The keyword itself.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
keyworddefs_name_idx	name	unique	-

The "keywords" table

Bugs may have keywords. This table defines which bugs have which keywords. The keywords are defined in the keyworddefs table.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	The bug (foreign key bugs.bug_id)
keywordid	smallint	0	-	The keyword ID (foreign key keyworddefs.id)

Indexes:

Name	Fields	Properties	Remarks
keywords_bug_id_idx	bug_id, keywordid	unique	-
keywords_keywordid_idx	keywordid		-

The "logincookies" table

Bugzilla generates a cookie each time a user logs in, and uses it for subsequent authentication. The cookies generated are stored in this table. For more information, see the notes on authentication.

Field	Type	Default	Properties	Remarks

cookie	varchar(16)	''	-	The cookie
ipaddr	varchar(40)	''	-	The CGI REMOTE_ADDR for this login.
lastused	datetime	0000-00-00 00:00:00	-	The timestamp of this login.
userid	mediumint	0	-	The user id; (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	cookie	unique	-
logincookies_lastused_idx	lastused		-

The "longdescs" table

Long bug descriptions.

Field	Type	Default	Properties	Remarks
already_wrapped	tinyint	0	-	Non-zero if this comment is word-wrapped in the database (and so should not be wrapped for display).
bug_id	mediumint	0	-	the bug (foreign key bugs.bug_id)
bug_when	datetime	0000-00-00 00:00:00	-	when the text was added
comment_id	mediumint	None	auto_increment	A unique ID for this comment.
extra_data	varchar(255)	None	null	Used in conjunction with longdescs.type to provide the variable data in localized text of an automatic comment. For instance, a duplicate bug number.
isprivate	tinyint	0	-	Non-zero if this comment is "private", i.e. only visible to members of the "insider" group.

thetext	mediumtext	''	-	the text itself.
type	smallint	0	-	The type of a comment, used to identify and localize the text of comments which are automatically added by Bugzilla. 0 for a normal comment. 1 for a comment marking this bug as a duplicate of another. 2 for a comment marking another bug as a duplicate of this. 3 for a comment recording a transition to NEW by voting. 4 for a comment recording that this bug has been moved.
who	mediumint	0	-	the user who added this text (foreign key profiles.userid)
work_time	decimal(5,2)	0.0	-	Number of hours worked on this bug (for time tracking purposes).

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	comment_id	unique	-
longdescs_bug_id_idx	bug_id		-
longdescs_bug_when_idx	bug_when		-
longdescs_who_idx	who, bug_id		-

The "milestones" table

Development milestones.

Field	Type	Default	Properties	Remarks
id	mediumint	None	auto_increment	A unique numeric ID
product_id	smallint	0	-	The product (foreign key products.id)
sortkey	smallint	0	-	A number used for sorting milestones for a given product.
value	varchar(20)	''	-	The name of the milestone (e.g. "3.1 RTM", "0.1.37", "tweakfor BigCustomer", etc).

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
milestones_product_id_idx	product_id, value	unique	-

The "namedqueries" table

Named queries.

Field	Type	Default	Properties	Remarks
id	mediumint	None	auto_increment	A unique number identifying this query.
name	varchar(64)	''	-	The name of the query.
query	mediumtext	''	-	The query (text to append to the query page URL).
query_type	tinyint	0	-	1 (LIST_OF_BUGS) if the query is simply a list of bug IDs, 0 (QUERY_LIST) if it is a genuine query.
userid	mediumint	0	-	The user whose query this is (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
namedqueries_userid_idx	userid, name	unique	-

The "namedqueries_link_in_footer" table

Controls whether a named query appears in a given user's navigation footer.

Field	Type	Default	Properties	Remarks

namedquery_id	mediumint	0	-	The query id (foreign key namedqueries.id).
user_id	mediumint	0	-	The user id (foreign key profiles.userid).

Indexes:

Name	Fields	Properties	Remarks
namedqueries_link_in_footer_id_idx	namedquery_id, user_id	unique	-
namedqueries_link_in_footer_userid_idx	user_id		-

The "namedquery_group_map" table

Controls whether a named query is shared with other users (other members of a group).

Field	Type	Default	Properties	Remarks
group_id	mediumint	0	-	The group id (foreign key groups.id).
namedquery_id	mediumint	0	-	The query id (foreign key namedqueries.id).

Indexes:

Name	Fields	Properties	Remarks
namedquery_group_map_group_id_idx	group_id		-
namedquery_group_map_namedquery_id_idx	namedquery_id	unique	-

The "op_sys" table

The possible values of the "operating system" field of a bug.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.

isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field
visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
op_sys_sortkey_idx	sortkey, value		-
op_sys_value_idx	value	unique	-
op_sys_visibility_value_id_idx	visibility_value_id		-

The "priority" table

The possible values of the "priority" field of a bug.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.
isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field
visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
priority_sortkey_idx	sortkey, value		-
priority_value_idx	value	unique	-
priority_visibility_value_id_idx	visibility_value_id		-

The "products" table

One row for each product. See the notes on products.

Field	Type	Default	Properties	Remarks
classification_id	smallint	1	-	The classification ID (foreign key classifications.id).
defaultmilestone	varchar(20)	---	-	The default milestone for a new bug (foreign key milestones.value)
description	mediumtext	None	null	The description of the product
disallownew	tinyint	0	-	New bugs can only be created for this product if this is 0.
id	smallint	None	auto_increment	The product ID.
maxvotesperbug	smallint	10000	-	Maximum number of votes which a bug may have.
milestoneurl	tinytext	''	-	The URL of a document describing the product milestones.
name	varchar(64)	''	-	The product name.
votesperuser	smallint	0	-	Total votes which a single user has for bugs of this product.
votestoconfirm	smallint	0	-	How many votes are required for this bug to become NEW.

Indexes:

Name	Fields	Properties	Remarks

PRIMARY	id	unique	-
products_name_idx	name	unique	-

The "profile_setting" table

User preference settings.

Field	Type	Default	Properties	Remarks
setting_name	varchar(32)	''	-	The name of the setting (foreign key setting.name).
setting_value	varchar(32)	''	-	The value (foreign key setting_value.value).
user_id	mediumint	0	-	The user (foreign key profiles.userid).

Indexes:

Name	Fields	Properties	Remarks
profile_setting_value_unique_idx	user_id, setting_name	unique	-

The "profiles" table

Describes Bugzilla users. One row per user.

Field	Type	Default	Properties	Remarks
cryptpassword	varchar(128)	None	null	The user's password. The Perl function `crypt` is used.
disable_mail	tinyint	0	-	1 to disable all mail to this user; 0 for mail to depend on the per-user email settings in email_setting.
disabledtext	mediumtext	''	-	If non-empty, indicates that this account has been disabled and gives a reason.
extern_id	varchar(64)	None	null	The ID for environmental authentication (see the notes on authentication).
login_name	varchar(255)	''	-	The user's email address. Used when logging in or providing mailto: links.

mybugslink	tinyint	1	-	indicates whether a "My Bugs" link should appear at the bottom of each page.
realname	varchar(255)	''	-	The user's real name.
userid	mediumint	None	auto_increment	A unique identifier for the user. Used in other tables to identify this user.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	userid	unique	-
profiles_login_name_idx	login_name	unique	-
profiles_extern_id_idx	extern_id	unique	-

The "profiles_activity" table

This table is for recording changes to the profiles table. Currently it only records changes to group membership made with editusers.cgi. This allows the administrator to track group inflation. There is currently no code to inspect this table; only to add to it.

Field	Type	Default	Properties	Remarks
fieldid	mediumint	0	-	The ID of the changed field (foreign key fielddefs.id)
newvalue	tinytext	None	null	The new value.
oldvalue	tinytext	None	null	The old value
profiles_when	datetime	0000-00-00 00:00:00	-	When it was changed
userid	mediumint	0	-	The profile which has changed (foreign key profiles.userid)
who	mediumint	0	-	The user who changed it (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
profiles_activity_userid_idx	userid		-

profiles_activity_profiles_when_idx	profiles_when		-
profiles_activity_fieldid_idx	fieldid		-

The "quips" table

A table of quips.

Field	Type	Default	Properties	Remarks
approved	tinyint	1	-	1 if this quip has been approved for display, 0 otherwise.
quip	mediumtext	''	-	The quip itself.
quipid	mediumint	None	auto_increment	A unique ID.
userid	mediumint	None	null	The user who added this quip (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	quipid	unique	-

The "rep_platform" table

The possible values of the "platform" field of a bug.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.
isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field

visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
rep_platform_sortkey_idx	sortkey, value		-
rep_platform_value_idx	value	unique	-
rep_platform_visibility_value_id_idx	visibility_value_id		-

The "resolution" table

The possible values of the "resolution" field of a bug.

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	a unique ID.
isactive	tinyint	1	-	1 if this value is available in the user interface, 0 otherwise
sortkey	smallint	0	-	A number used to determine the order in which values are shown.
value	varchar(64)	''	-	A possible value of the field
visibility_value_id	smallint	None	null	If set, this value is only available if the chooser field (identified by fielddefs.value_field_id) has the value with this ID. Foreign key <field>.id, for example products.id or cf_<field>.id.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-

resolution_sortkey_idx	sortkey, value		-
resolution_value_idx	value	unique	-
resolution_visibility_value_id_idx	visibility_value_id		-

The "series" table

Properties of the time-series datasets available (e.g. for plotting charts). See the notes on charts.

Field	Type	Default	Properties	Remarks
category	smallint	0	-	The series category. (foreign key series_categories.id)
creator	mediumint	None	null	The user who created this series (foreign key profiles.userid). NULL if this series is created by checksetup when first installing Bugzilla.
frequency	smallint	0	-	The period between data samples for this series, in days.
last_viewed	datetime	None	null	The time at which this dataset was last viewed.
name	varchar(64)	''	-	The series name.
is_public	tinyint	0	-	1 if the series is visible to all users, 0 otherwise.
query	mediumtext	''	-	a snippet of CGI which specifies a subset of bugs, as for query.cgi
series_id	mediumint	None	auto_increment	A unique ID.
subcategory	smallint	0	-	The series subcategory. (foreign key series_categories.id)

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	series_id	unique	-
series_creator_idx	creator, category, subcategory, name	unique	-

The "series_categories" table

Field	Type	Default	Properties	Remarks
id	smallint	None	auto_increment	A unique ID.
name	varchar(64)	''	-	The category name.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
series_categories_name_idx	name	unique	-

The "series_data" table

Data for plotting time-series charts. See the notes on charts.

Field	Type	Default	Properties	Remarks
series_date	datetime	0000-00-00 00:00:00	-	The time point at which this datum was collected.
series_id	mediumint	0	-	The series ID. (foreign key series.series_id)
series_value	mediumint	0	-	The number of bugs in the dataset at this time point.

Indexes:

Name	Fields	Properties	Remarks
series_data_series_id_idx	series_id, series_date	unique	-

The "setting" table

Identifies the set of user preferences.

Field	Type	Default	Properties	Remarks

default_value	varchar(32)	''	-	the value of this setting which will apply to any user who does not change it.
is_enabled	tinyint	1	-	1 if users are able to change this setting; 0 if it is automatic.
name	varchar(32)	''	-	The name of the setting.
subclass	varchar(32)	None	null	The name of the Perl subclass (of Setting) to which this setting applies.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	name	unique	-

The "setting_value" table

Possible values for user preferences.

Field	Type	Default	Properties	Remarks
name	varchar(32)	''	-	The setting name. (foreign key setting.name)
sortindex	smallint	0	-	A number used to determine the order in which setting values are shown
value	varchar(32)	''	-	The setting value

Indexes:

Name	Fields	Properties	Remarks
setting_value_ns_unique_idx	name, sortindex	unique	-
setting_value_nv_unique_idx	name, value	unique	-

The "status_workflow" table

Identifies allowable workflow transitions.

Field	Type	Default	Properties	Remarks
new_status	smallint	0	-	The new bug status (foreign key bug_status.id)
old_status	smallint	None	null	The old bug status, None for bug creation (foreign key bug_status.id)
require_comment	tinyint	0	-	1 if this transition requires a comment; 0 otherwise.

Indexes:

Name	Fields	Properties	Remarks
status_workflow_idx	old_status, new_status	unique	-

The "tokens" table

Tokens are sent to users to track activities such as creating new accounts and changing email addresses or passwords. They are also sent to browsers and used to track workflow, to prevent security problems (e.g. so that one can only delete groups from a session last seen on a group management page).

Field	Type	Default	Properties	Remarks
eventdata	tinytext	None	null	The expected event, for a session token.
issuedate	datetime	0000-00-00 00:00:00	-	The date at which the token was issued
token	varchar(16)	''	-	The token itself.
tokentype	varchar(8)	''	-	The type of the token. Possible values: 'account' when creating a new user account, 'emailold' and 'emailnew' when changing email address, 'password' when changing a password, or 'session' for a session token.
userid	mediumint	None	null	The user to whom the token was issued. (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks

PRIMARY	token	unique	-
tokens_userid_idx	userid		-

The "ts_error" table

A log of errors from TheSchwartz asynchronous job-queueing system. Rows are aged out of this table after seven days.

Field	Type	Default	Properties	Remarks
error_time	int	0	-	The time at which the error occurred.
funcid	int	0	-	The function ID. Foreign key ts_funcmap.funcid.
jobid	int	0	-	The job ID. Foreign key ts_job.jobid
message	varchar(255)	''	-	The error message.

Indexes:

Name	Fields	Properties	Remarks
ts_error_funcid_idx	funcid, error_time		-
ts_error_jobid_idx	jobid		-
ts_error_error_time_idx	error_time		-

The "ts_exitstatus" table

A log of job completions from TheSchwartz asynchronous job-queueing system.

Field	Type	Default	Properties	Remarks
completion_time	int	None	null	The time at which the job finished.
delete_after	int	None	null	A time after which this row can be deleted.
funcid	int	0	-	The function ID. Foreign key ts_funcmap.funcid.

jobid	int	None	auto_increment	The job ID. Foreign key ts_job.jobid
status	smallint	None	null	The exit status. 0 for success.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	jobid	unique	-
ts_exitstatus_delete_after_idx	delete_after		-
ts_exitstatus_funcid_idx	funcid		-

The "ts_funcmap" table

The table of functions for TheSchwartz asynchronous job-queueing system.

Field	Type	Default	Properties	Remarks
funcid	int	None	auto_increment	A unique ID.
funcname	varchar(255)	''	-	A unique function name, also known as an ability or a worker class name.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	funcid	unique	-
ts_funcmap_funcname_idx	funcname	unique	-

The "ts_job" table

The job queue managed by TheSchwartz asynchronous job-queueing system.

Field	Type	Default	Properties	Remarks

arg	longblob	None	null	State data for the job, stored as a frozen reference.
coalesce	varchar(255)	None	null	A string used to indicate jobs which can be usefully pipelined by a single worker.
funcid	int	0	-	The function ID. Foreign key ts_funcmap.funcid.
grabbed_until	int	0	-	Set while a worker is attempting this job; do not retry this job until this is in the past.
insert_time	int	None	null	not used.
jobid	int	None	auto_increment	A unique ID.
priority	smallint	None	null	Not used.
run_after	int	0	-	A timestamp before which the job should not be run.
uniqkey	varchar(255)	None	null	An arbitrary unique reference.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	jobid	unique	-
ts_job_coalesce_idx	coalesce, funcid		-
ts_job_funcid_idx	funcid, uniqkey	unique	-
ts_job_run_after_idx	run_after, funcid		-

The "ts_note" table

Notes on jobs for TheSchwartz asynchronous job-queueing system. Apparently not used.

Field	Type	Default	Properties	Remarks
jobid	int	0	-	The job ID. Foreign key ts_job.jobid
notekey	varchar(255)	None	null	Not used.

value	longblob	None	null	Not used.

Indexes:

Name	Fields	Properties	Remarks
ts_note_jobid_idx	jobid, notekey	unique	-

The "user_group_map" table

This table records which users are members of each group, or can "bless" each group. See the notes on groups.

Field	Type	Default	Properties	Remarks
grant_type	tinyint	0	-	0 if this membership or privilege is explicit. 1 if it is derived from a group hierarchy (see the group_group_map table). 2 if it results from matching a regular expression (see groups.userregexp).
group_id	mediumint	0	-	The group. (foreign key groups.id)
isbless	tinyint	0	-	0 if this row records group membership; 1 if this row records group "bless" privilege.
user_id	mediumint	0	-	The user. (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
user_group_map_user_id_idx	user_id, group_id, grant_type, isbless	unique	-

The "versions" table

Product versions.

Field	Type	Default	Properties	Remarks
id	mediumint	None	auto_increment	A unique numeric ID

product_id	smallint	0	-	The product (foreign key products.id)
value	varchar(64)	''	-	The name of the version

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
versions_product_id_idx	product_id, value	unique	-

The "votes" table

votes.

Field	Type	Default	Properties	Remarks
bug_id	mediumint	0	-	The bug (foreign key bugs.bug_id)
vote_count	smallint	0	-	How many votes.
who	mediumint	0	-	The user (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
votes_bug_id_idx	bug_id		-
votes_who_idx	who		-

The "watch" table

watchers.

Field	Type	Default	Properties	Remarks

watched	mediumint	0	-	The watched user (foreign key profiles.userid)
watcher	mediumint	0	-	The watching user (foreign key profiles.userid)

Indexes:

Name	Fields	Properties	Remarks
watch_watched_idx	watched		-
watch_watcher_idx	watcher, watched	unique	-

The "whine_events" table

One row for each regular whine event. See the notes on whining.

Field	Type	Default	Properties	Remarks
body	mediumtext	None	null	Text to appear in the body of the whine emails before the bugs table.
id	mediumint	None	auto_increment	The whine event ID, used to identify this event.
owner_userid	mediumint	0	-	The user ID of the whine owner (foreign key profiles.userid). Must match namedqueries.userid for the queries associated with this event (whine_queries.query_name).
subject	varchar(128)	None	null	The Subject of the whine emails.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-

The "whine_queries" table

See the notes on whining.

Field	Type	Default	Properties	Remarks
eventid	mediumint	0	-	The whine event ID (foreign key whine_events.id).
id	mediumint	None	auto_increment	A unique ID for this query.
onemailperbug	tinyint	0	-	1 if a separate email message should be sent for each bug matching the query; 0 if a single email should be sent covering all the bugs.
query_name	varchar(64)	''	-	The query name (foreign key namedqueries.name).
sortkey	smallint	0	-	A key to order the queries for a given event ID.
title	varchar(128)	''	-	The title displayed for this query in the message.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
whine_queries_eventid_idx	eventid		-

The "whine_schedules" table

See the notes on whining.

Field	Type	Default	Properties	Remarks
eventid	mediumint	0	-	The whine event ID (foreign key whine_events.id).
id	mediumint	None	auto_increment	a unique ID for this whine schedule.
mailto	mediumint	0	-	Either a user ID (foreign key profiles.userid) or group ID (foreign key groups.id) identifying the user or users to whom to send whine messages.
mailto_type	smallint	0	-	0 if the mailto field is a user ID, 1 if it is a group ID.
run_day	varchar(32)	None	null	The day on which this whine should run. 'All' means every day. 'MF' means Monday to Friday inclusive. A three letter weekday abbreviation (e.g. "Mon", "Thu") means only on that day. An integer indicates a particular day of the month.

				'last' means the last day of the month.
run_next	datetime	None	null	The time and date at which the whine should next be run. NULL if the whine has been changed and not rescheduled yet.
run_time	varchar(32)	None	null	The time at which this whine should run. An integer indicates an hour of the day. An interval (e.g. "15min", "30min") indicates that the whine should run repeatedly at that interval.

Indexes:

Name	Fields	Properties	Remarks
PRIMARY	id	unique	-
whine_schedules_eventid_idx	eventid		-
whine_schedules_run_next_idx	run_next		-

4. Bugzilla History

Bugzilla releases

This table gives the dates of all the Bugzilla releases since 2.0.

Date	Release	Notes
1998-09-19	2.0	Not described in this document.
1999-02-10	2.2	Not described in this document.
1999-04-30	2.4	Not described in this document.
1999-08-30	2.6	Not described in this document.
1999-11-19	2.8	Not described in this document.
2000-05-09	2.10	Not described in this document.
2001-04-27	2.12	Not described in this document.

2001-08-29	2.14	Not described in this document.
2002-01-05	2.14.1	A security patch release. Not described in this document.
2002-05-10	2.16rc1	A release candidate. Not described in this document.
2002-06-07	2.16rc2	A release candidate. Not described in this document.
2002-06-07	2.14.2	A security patch release. Not described in this document.
2002-07-28	2.16	Not described in this document.
2002-07-28	2.14.3	A security patch release. Not described in this document.
2002-09-30	2.16.1	A security patch release. Not described in this document.
2002-09-30	2.14.4	A security patch release. Not described in this document.
2002-11-25	2.17.1	A development release. Not described in this document.
2003-01-02	2.16.2	A security patch release. Not described in this document.
2003-01-02	2.14.5	A security patch release. Not described in this document.
2003-01-02	2.17.3	A development release. Not described in this document.
2003-04-25	2.16.3	A security patch release. Not described in this document.
2003-04-25	2.17.4	A development release. Not described in this document.
2003-11-03	2.17.5	A development release. Not described in this document.
2003-11-03	2.16.4	A security patch release Not described in this document.
2003-11-10	2.17.6	A development release. Not described in this document.
2004-03-03	2.16.5	A security patch release Not described in this document.
2004-03-03	2.17.7	A development release. Not described in this document.
2004-07-10	2.16.6	A security patch release Not described in this document.
2004-07-10	2.18rc1	A release candidate. Not described in this document.
2004-07-28	2.18rc2	A release candidate. Not described in this document.

2004-10-24	2.16.7	A security patch release Not described in this document.
2004-10-24	2.18rc3	A release candidate. Not described in this document.
2004-10-24	2.19.1	A development release. Not described in this document.
2005-01-15	2.16.8	A security patch release Not described in this document.
2005-01-15	2.18	Not described in this document.
2005-01-15	2.19.2	A development release. Not described in this document.
2005-05-12	2.16.9	A security patch release Not described in this document.
2005-05-12	2.18.1	A security patch release Not described in this document.
2005-05-12	2.19.3	A development release. Not described in this document.
2005-05-19	2.16.10	A security patch release Not described in this document.
2005-07-08	2.18.2	A security patch release Not described in this document.
2005-07-08	2.20rc1	A release candidate Not described in this document.
2005-07-09	2.18.3	A security patch release Not described in this document.
2005-08-08	2.20rc2	A release candidate Not described in this document.
2005-10-01	2.18.4	A security patch release Not described in this document.
2005-10-01	2.20	Not described in this document.
2005-10-01	2.21.1	A development release. Not described in this document.
2006-02-21	2.16.11	A security patch release Not described in this document.
2006-02-21	2.18.5	A security patch release Not described in this document.
2006-02-21	2.20.1	A security patch release Not described in this document.
2006-02-21	2.22rc1	A release candidate Not described in this document.
2006-04-23	2.20.2	A security patch release Not described in this document.
2006-04-23	2.22	Not described in this document.

2006-04-23	2.23.1	A development release. Not described in this document.
2006-07-09	2.23.2	A development release. Not described in this document.
2006-10-15	2.18.6	A security patch release Not described in this document.
2006-10-15	2.20.3	A security patch release Not described in this document.
2006-10-15	2.22.1	A security patch release Not described in this document.
2006-10-15	2.23.3	A development release Not described in this document.
2007-02-02	2.20.4	A security patch release Not described in this document.
2007-02-02	2.22.2	A security patch release Not described in this document.
2007-02-02	2.23.4	A development release Not described in this document.
2007-02-26	3.0rc1	A release candidate Not described in this document.
2007-05-09	3.0	Not described in this document.
2007-08-23	2.20.5	A security patch release Not described in this document.
2007-08-23	2.22.3	A security patch release Not described in this document.
2007-08-23	3.0.1	A security patch release Not described in this document.
2007-08-23	3.1.1	A development release Not described in this document.
2007-09-19	3.0.2	A security patch release Not described in this document.
2007-09-19	3.1.2	A development release Not described in this document.
2008-01-09	3.0.3	A patch release Not described in this document.
2008-02-02	3.1.3	A development release Not described in this document.
2008-05-04	2.20.6	A security patch release Not described in this document.
2008-05-04	2.22.4	A security patch release Not described in this document.
2008-05-04	3.0.4	A security patch release Not described in this document.
2008-05-04	3.1.4	A development release Not described in this document.

2008-08-12	2.22.5	A security patch release Not described in this document.
2008-08-12	3.0.5	A security patch release Not described in this document.
2008-08-12	3.2rc1	A release candidate Not described in this document.
2008-11-07	2.20.7	A security patch release Not described in this document.
2008-11-07	2.22.6	A security patch release Not described in this document.
2008-11-07	3.0.6	A security patch release Not described in this document.
2008-11-07	3.2rc2	A release candidate Not described in this document.
2008-11-30	3.2	Not described in this document.
2009-01-06	3.3.1	A development release Not described in this document.
2009-02-03	2.22.7	A security patch release Not described in this document.
2009-02-03	3.0.7	A security patch release Not described in this document.
2009-02-03	3.2.1	A security patch release Not described in this document.
2009-02-03	3.0.8	A security patch release Not described in this document.
2009-02-03	3.2.2	A security patch release Not described in this document.
2009-02-03	3.3.2	A development release Not described in this document.
2009-02-03	3.3.3	A security patch reease Not described in this document.
2009-03-31	3.2.3	A security patch release Not described in this document.
2009-03-31	3.3.4	A development release Not described in this document.
2009-07-08	3.2.4	A security patch release Not described in this document.
2009-07-08	3.4rc1	A development release Not described in this document.
2009-07-28	3.4	Not described in this document.
2009-08-01	3.4.1	A security patch release Not described in this document.
2009-09-11	3.0.9	A security patch release Not described in this document.

2009-09-11	3.2.5	A security patch release Not described in this document.
2009-09-11	3.4.2	A security patch release

Bugzilla Schema Changes

5. Example queries

To select bug number *n*:

```
select * from bugs where bug_id = n
```

To get a complete list of user ids and email addresses:

```
select userid, login_name from profiles
```

To get the email address of user *n*:

```
select login_name from profiles where userid = n
```

To get the set of cc addresses of bug *n*:

```
select login_name from cc, profiles where cc.bug_id = n and profiles.userid = cc.who
```

To select the long descriptions of bug *n*, together with the name and email address of the commenters:

```
select profiles.login_name, profiles.realname, longdescs.bug_when, longdescs.thetext from longdescs, profiles where
profiles.userid = longdescs.who and longdescs.bug_id = n order by longdescs.bug_when
```

To find out the groups of user *n*:

```
select group_id from user_group_map where userid = n and isbless=0
```

A. References

B. Document History

2000- NB Created.

11-14		
2001-03-02	RB	Transferred copyright to Perforce under their license.
2001-04-06	NB	Added sample queries.
2001-09-12	NB	Updated to reflect schema updates in Bugzilla 2.12 and 2.14
2002-01-31	NB	Added notes on Bugzilla 2.14.1.
2002-05-31	NB	Updated for Bugzilla 2.16 (based on 2.16rc1).
2002-09-26	NB	Updated for Bugzilla 2.16/2.14.2/2.14.3.
2002-10-04	NB	Added notes on Bugzilla 2.14.4 and 2.16.1, and on identical schemas.
2003-05-14	NB	Added extensive notes on schema changes, in section 2.
2003-06-06	NB	Added table of Bugzilla releases showing release date and support status.
2003-06-06	NB	Added notes on schema changes in 2.17.x.
2003-06-13	NB	Added first cut at description of new Bugzilla tables.
2003-06-27	NB	Added more on recent schema changes. Colour-coded all schema changes.
2003-07-09	NB	Completely changed the way this document is produced. The schema tables themselves are now created and coloured automatically by querying MySQL.
2003-11-04	NB	Add Bugzilla 2.16.4 and 2.17.5.
2003-11-10	NB	Add Bugzilla 2.17.6.
2004-03-19	NB	Add Bugzilla 2.17.7; improve documentation of the groups system; improve automated schema change descriptions.
2004-	NB	Add documentation of the flags system, the time series system, and the time tracking system.

03-26		
2004-04-30	NB	Correct some documentation of the time series system based on feedback from the author.
2004-07-14	NB	Add 2.16.6 and 2.18rc1.
2004-07-28	NB	Add 2.18rc2.
2004-11-11	NB	Add 2.16.7, 2.18rc3, 2.19.1. Change document-generation code to improve colouring, link consistency, control, and robustness.
2004-11-12	NB	Turn into CGI, using schemas stored in Python pickles.
2004-11-13	NB	Add 2.0, 2.2, 2.4, 2.6. 2.8, for completeness.
2004-12-03	NB	Add notes on quips and a few missing foreign key links.
2005-01-18	NB	Add 2.16.8, 2.18, and 2.19.2.
2005-05-19	NB	Add 2.16.9, 2.16.10, 2.18.1, and (preliminarily) 2.19.3.
2005-09-15	NB	Add 2.18.2, 2.18.3, 2.20rc1, 2.20rc2, and complete remarks for 2.19.3.
2005-10-03	NB	Add 2.18.4, 2.20, 2.21.1
2006-05-18	NB	Add 2.16.11, 2.18.5, 2.20.1, 2.22rc1, 2.20.2, 2.22, 2.23.1.
2006-10-31	NB	Add recent releases, to 2.18.6, 2.20.3, 2.22.1, 2.23.3.
2007-05-11	NB	Add recent releases 2.20.4, 2.22.2, 2.23.4, 3.0rc1, 3.0.
2008-02-29	NB	Add recent releases 3.0.1, 3.0.2, 3.0.3, 3.1.1, 3.1.2, 3.1.3.
2009-07-31	NB	Add recent releases 2.20.7, 2.22.5, 2.22.6, 2.22.7, 3.0.5, 3.0.6, 3.0.7, 3.0.8, 3.2rc1, 3.2rc2, 3.2, 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.4rc1, and 3.4.

Generated at 2015-10-26 17:37:29
by `//info.ravenbrook.com/project/bugzilla-schema/master/make_schema_doc.py#1`
from `//info.ravenbrook.com/project/bugzilla-schema/master/schema_remarks.py#4`
