Planforge

INTRODUCTION TO THE PLANFORGE DATABASE SCHEMA

KEY DESIGN GOALS

Planforge's database schema was designed with a number of key design goals in mind, mainly being open and accessible.

Open and Accessible

The most important goal was to design the database schema in a way that it is easily accessible. The primary reason for this being that even the best internal reporting tool will not be sufficient for all applications. Especially, executive reporting is often done using one, standardized reporting/business intelligence tool that is used across all data being gathered in the enterprise or at least a specific division. In order to succeed on this design goal, the following requirements had to be met:

- Consistent structure and human-readable naming in order that a 3rd party can understand and use the database schema
- No or very limited use of proprietary or binary data types – stick to standard SQL as far as possible
- No incremental storage (e.g., for versioning), since it makes external querying of the database problematic and sometimes nearly impossible Finally, the database schema would have to be open, i.e., documented.

Fast Access to Monitoring Information

If you take a closer look at some of the tables you will seethat quite a bit of information is stored redundantly. This might not be the "good" thing to do on paper, but it is the only possibility in order to provide a real-time view on certain monitoring information (performance/scalability-wise). This information especially includes (mostly hierarchically) summed-up information, e.g., the sum of actual effort tracked for an assignment (from its work records), the sum of actual effort tracked for an activity (from its assignments), or the sum of actual effort tracked for a super-activity (from its sub-activities). Summing all this information up at run-time would simply not scale as soon as there are more than just a few projects stored in the repository.

LIMITATIONS AND DANGERS OF DIRECT ACCESS

Even though Planforge's database schema was designed to be open, it does not mean that it is always a good idea to access the database directly

IMPORTANT: READ-ONLY External Access

Planforge currently uses a so-called “hi/lo generator” for creating new, unique object IDs for business objects (such as, e.g., projects or activities) to be stored in the database. In addition, Planforge caches information in memory – including potential session-level database locks that are (mainly for performance and scalability reasons) not always persisted in the database.

These two things said, it is not a good idea to write information directly into an Planforge – unless you know 100% what you are doing. In fact, we strongly discourage anyone from writing information directly into the database and if you do so your installation will no longer be covered by any support contract issued by the onepoint PROJECTS GmbH.

If you need to write information into the database the good way to do this is to use the Planforge Web Services or REST API available in Planforge 10 or later. This is also the way the Planforge professional services team solves issues where it is required to get information from external data sources into the Planforge project repository, e.g., if an external time tracking system is to be integrated.

Planforge: The Most Important Tables
- op_projectnode
- op_resource
- op_projectplan
- op_activity
- op_assignment
- op_workrecord

Bypassing the Access Control System (!)

Planforge protects stored data via a fine-granular access control system based on users, user groups and role-based access permissions. By accessing an Planforge database instance directly you need to be aware of the fact
that you are bypassing this access control system.

Unresolved Language Resource References

Another less important, but still “good to know” fact is that you can encounter object names in the database that are in fact unresolved language resources. These language resources have the format “${ResourceName}”. A good example for this is the porfolio with the name “${RootProjectPortfolioName}” that refers to the “Base Project Portfolio” in the application.

THE CORE DATABASE SCHEMA

Planforge creates quite a few tables in its database schema. However, only a limited number of tables deal with the core data being project and activity name, resource assignments, dates, effort, duration and costs.

Unique Object Identifiers and Other Common Columns

Every object in an Planforge repository has a unique object ID ("op_id"). In addition, it stores the objects creation date ("op_created") and an optional modification date ("op_modified"). These three columns are present in all tables that we will describe in the following sections.

Organizational Data: Projects, Templates and Portfolios

Portfolios, projects and templates are stored in a single table called "op_projectnode". Note that the project plan is stored separately and only references the project node it belongs to.

The most important field of the op_projectnode table is the smallint field "op_type". It denotes whether the project node stores a portfolio, a project or a template:

- 1: Portfolio
- 2: (Reserved for future use)
- 3: Project
- 4: Template

Other important fields of the project node are:

• op_name (varchar, mandatory, unique): The unique name of the project, template or portfolio
• op_description (varchar): An optional description
• op_start (date): Intended start date of the project
• op_finish (date): Intended finish date of the project
• op_budget (double): Intended budget for the project
• op_priority (integer): Priority of the project
• op_probability (integer): Probability of the project being started
• op_archived (boolean): If true this project is only displayed in the project administration and hidden in all other views
• op_supernode (bigint): Object ID of the super project node (this field creates the portfolio hierarchy)

Organizational Data: Resources and Pools

Individual resources and resource pools are stored in the op_resource table. The hierarchy is constructed the same way as for project nodes (via a op_resourcepool object ID relationship). The most important field of the op_resource table is the smallint field “op_type”. It denotes whether the resource node stores a resource, a resource pool or a collection resource:

• 0: Resource
• 1: Resource Pool
• 2: Collection Resource

Other important fields of the resource node are:

• op_name (varchar, mandatory, unique): The unique name of the resource pool
• op_description (varchar): An optional description
• op_hourlyrate (double): The (default) internal hourly rate for the resource pool
• op_externalrate (double): The (default) external hourly rate for the resource pool
• op_resourcepool (bigint): Object ID of the super resource pool
• op_user (bigint): Object ID of the linked user
• op_available (double): The availability of the resource

Note that internal hourly rates are used to calculate personnel costs and external hourly rates to calculate

proceeds (currently there are only personnel proceeds which will most likely change in the near future). Pool-inheritance is already reflected in the stored data.

The Project Plan: Activities and Assignments

All project plans are stored in the op_projectplan table. Every project node of type project or template can have an associated project plan. The most relevant fields of the op_projectplan table are as follows:

• op_start (date): Minimum start date of all activities in the project plan
• op_finish (date): Maximum finish date of all activities in the project plan
• op_projectnode (bigint): Object ID of the project node the plan belongs to

1.1.1.1 Activities

The individual activities of a project plan are stored as activities in the op_activity table. Each activity belongs exactly to one project plan (op_projectplan). The most important field of the op_activity table is the smallint field “op_type”. It denotes which kind of activity is stored in the table:

• 0: Activity
• 1: Collection Activity
• 2: Milestone
• 3: Task
• 4: Collection Task
• 5: Scheduled Task
• 6: Ad Hoc Task
• 7: Issue

Other important fields of the op_activity table:
• op_name (varchar): The name of the activity
• op_description (varchar): An optional description
• op_sequence (integer): The sequence number of the activity within the project plan
• op_outlinelevel (integer): The outline level of the activity (0 means it is a top-level activity)
• op_start (date): The start date of the activity
• op_finish (date): The finish date of the activity

• op_duration (double): The planned activity duration in working hours
• op_complete (double): The completeness of the activity (in percent, i.e., 100.0 means 100%)
• op_responsibleresource (bigint): Object ID of the responsible resource
• op_superactivity (bigint): Object ID of the super activity of this activity (null if this activity is a top-level activity)
• op_deleted (Boolean): If this is true this is an activity that has been marked as deleted
• op_baseeffort (double): Planned effort in working hours
• op_base…costs (double): Planned costs (where op_basepersonnelcosts is calculated based on internal hourly rates and resource assignments)
• op_baseproceeds (double): Planned personnel proceeds (calculated based on external hourly rates and resource assignments)
• op_actualeffort (double): Actual, tracked effort in working hours
• op_actual…costs (double): Actual, tracked costs (where op_actualpersonnelcosts is calculated based on internal hourly rates and resource assignments)
• op_actualproceeds (double): Actual, tracked personnel proceeds (calculated based on external hourly rates and resource assignments)

In addition, you will find op_remaining… fields analogous to the various op_base… and op_actual… fields. These fields contain effort-to-complete/cost-to-complete estimations that have been either calculated from the op_complete field or reflect summed-up estimations from assignments/work records (see below) – based on the currently set calculation mode of the project (effort-based vs. independent planning). Note that the activity hierarchy can be reconstructed in two ways:

• Either by using the op_superactivity field (typically preferable)
• Or by using the combination of op_sequence and op_outlinelevel

Note also that you can typically ignore activities marked as deleted (op_deleted). There is a technical reason behind sometimes keeping an activity in a plan even though it was deleted.

Resource Assignments

Every “leaf-activity” in the project repository is allowed to have one or more resource assignments, i.e., every activity that has no sub-activities. These assignments are stored in the op_assignment table that in turn defines the following
important fields:

• op_assigned (double): Planned assignment of the resource in percent of working time per workday (100.0 being 100%)
• op_activity (bigint): The object ID of the activity the resource assignment belongs to

• op_resource (bigint): The object ID of the resource being assigned to the activity
• op_baseeffort (double): Planned effort in working hours for this resource (already taking the op_assigned value in consideration)
• op_basecosts (double): Planned personnel costs for this resource (based on planned effort and internal hourly rates)
• op_baseproceeds (double): Planned personnel proceeds for this resource (based on planned effort and external hourly rates)
• op_actualeffort (double): Actual, tracked effort in working hours for this resource (already taking the op_assigned value in consideration)
• op_actualcosts (double): Actual tracked personnel costs for this resource (based on planned effort and internal hourly rates)
• op_actualproceeds (double): Actual, tracked personnel proceeds for this resource (based on planned effort and external hourly rates)

Please note that the op_remaining… fields contain effort-tocomplete/cost-to-complete information analogous to activities, but on the assignment-level; cost fields on this level always refer to personnel costs.

1.1.1.2 Deprecated Tables

Although the Planforge database schema remained pretty stable over the last couple of years, we still had a number of smaller (automatic) schema migrations – especially, in order to respond to growing demands of our customers in the project resources and cost controlling areas. Due to this fact, there are a small number of tables you might find in your database schema that are deprecated, i.e., no longer in use by Planforge. When example is the opresourcepool table that once contained the resource pools that are now managed also in the opresource table together with other types of resources such as normal and collection resources.

The most “famous” deprecated tables are:

• op_object
• op_resourcepool
• op_workslip

If you come across one of these tables in a Planforge 10 database schame (or a later version) you can safely ignore these.