Roundtable White Paper

Managing Schema in Roundtable

Summary

This paper is intended to clarify the management of schema in Roundtable. It is assumed that you have a firm understanding of Roundtable but need extra assistance with schema management. It is not intended to replace the Roundtable User's Guide.

Definitions

PDBASE Object	- links to the database object
PFILE Object	- database table
PFIELD Object	- database field
Check Out	- Version an object
RTB	- Abbreviation for Roundtable
RT v8.3B02	- for Progress V8.2A - V8.3x

Top Rules (will be explained later in this document).

1. Never use schema load to "fix" a non-primary workspace. Schema load does NOT look at the physical database and then try to find corresponding object definitions in RTB to assign. It is designed to load in new changes - in other words, it creates new table or field objects, or checks out existing table or field objects in order to change them. This is extremely useful in your primary development environment to load in changes from the physical DB made in the data dictionary or loaded into your physical DB from a third party tool, such as Erwin.
2. ***Never*** try to fix schema problems by writing your own hacks to massage the Roundtable repository.
3. Always research the impact of assigning a previous version of a PDBASE or PFILE to a workspace before running the schema update.

Managing Schema

PDBASE Object - Logical and Physical Name

The PDBASE object name is used as the logical name for the database. The physical name is specified in the connection parameters for the PDBASE object and is used to connect the DB using a standard Progress "CONNECT" statement. You can change the physical connection parameters at any time, as they are not associated with a specific version of a PDBASE object. They are not magically entered by default when you define a PDBASE object.

If you leave the physical connection parameters blank, RTB does not try to connect to the physical database and will not complain about the missing connection. The physical database only needs to be connected when the standard Progress compile demands it (no different than if you were not running RTB).

"Double Assignment"

Schema objects maintain two assignments in RTB:
1. A PFILE or PFIELD object is assigned to a workspace (similar to PCODE objects).
2. A PFILE is assigned to zero or more PDBASE objects. A PFIELD is assigned to zero or more PFILE objects. In other words, you must assign a field to a table and a table to a database.

This "double assignment" is the source of much confusion to new Roundtable users. When you create a PFILE object in a workspace, you have only created a "free-floating" table. It does not exist in a database until you expand the database and assign the table to it. This leads to a FAQ, "I created schema but the Update Schema Process reports that there are no changes to apply to the physical database".

Database Versions

A version of a PDBASE has a "set" of PFILE objects assigned to it. You must check out the PDBASE object only if you wish to remove or add any of its PFILE objects. Making a change to a PFILE object does not affect the PDBASE object.

Table Versions

A version of a PFILE has a "set" of PFIELD objects assigned to it.   You must check out the PFILE object only if you wish to add or remove one or more of its PFIELD objects. Making a change to a PFIELD does not affect the PFILE.

Field Versions

A change to a PFIELD object is NOT a change to its table(s). The PFIELD object is versioned without having to check out a PFILE. This is because you are not changing the set of fields assigned to the PFILE.

How to Get in Trouble

If a version of a PDBASE object has three PFILE objects assigned to it, then those three PFILES must also be assigned to the workspace. If a version of a PFILE object has three PFIELD objects assigned to it, then those three PFIELD objects must also be assigned to the workspace. Since you cannot assign a PFILE to a PDBASE or a PFIELD to a PFILE unless it is already assigned to the workspace, this usually does not cause too much trouble. You can get confused if you assign previous versions of PDBASE or PFILE objects that do not have the proper PFILE or PFIELD objects assigned to the workspace.
For Example:

1. You create a PDBASE object called "sports" V010000 with three PFILE objects: "customer", "order", and "invoice".
2. You check out "sports" creating V010001 and remove the assignment of the "order" PFILE object.
3. Performa a schema update to apply the changes in RTB to the physical database.
4. Either remove the "order" PFILE object from the workspace or say "yes" when RTB asks if it should be removed from the workspace since the PDABSE object no longer uses it.
5. Complete the task in order to check in the PDBASE object. You now have V010001 of "sports" with two tables assigned to it: "customer" and "invoice".
6. ASSIGN V010000 of the "sports" PDBASE object to the workspace. This version of "sports" has three PFILE objects assigned to it: "customer", "order", and "invoice".
7. Do a schema update to update the physical DB with the changes made in RTB - IT WILL FAIL!

This example causes a failure because V010000 of the "sports" PDBASE object has the "order" PFILE object assigned it, but the "order" PFILE object was removed from the workspace when we removed it in V010001 of "sports". To recover, you simply have to assign the desired version of the "customer" PFILE object to the workspace.

The Schema Update Process

When you make changes to schema objects in RTB, you are making logical changes. The schema update process applies the logical changes made in RTB to the physical database. This allows you to make many changes logically over a long period, then apply them all at once to the physical database.

You cannot apply a change to a database that already exits, such as adding a table to a database when the table is already there (perhaps someone added it to the physical database using the data dictionary and also added it in RTB). The "skip" option is used when you want RTB to make the schema objects as being applied without actually touching the physical database.

IMPORTING

Schema is treated the same as code in Roundtable. Each workspace in Roundtable typically has its own set of application databases (see the "Extras" section in this document for information on how to work around this). This allows you to make many more changes in your development environment without affecting the testing environment.

The import process compares the versions of objects in a target workspace to the versions of objects in a defined base line (Release, task, etc) coming from a source workspace and builds an import list. It then assigns the desired versions of objects in the target workspace, bringing the target workspace up to the desired base line. When you bring over schema changes, you are bringing the logical changes in RTB into the target workspace. You will then have to run a schema update to update the physical database.

Compiling

There are two parts to compiling an object in RTB. The first half is the standard Progress compile, where Progress expects the tables and fields referenced in a program to be in the physical database. If you made a logical change in RTB (added a field) but did not run the Schema Update to apply the changes, the compile of a program that references this field will fail at this point.

After a successful Progress compile, the process moves to the XREF. Depending upon the XREF level, RTB expects to find logical objects for all schema referenced by the program. If a field was added to a database using the Data Dictionary but was not created or loaded into RTB, a level 5 XREF would fail the compile since the field object is not in RTB.

Extras

Fooling RTB into Using a Single Database for all Workspaces

It may be easier for you to use a single set of databases for all workspaces if you rarely make schema changes. Since the physical database and logical PDBASE object are separate, you can simply enter the same connection parameters for the database in each workspace.

* After importing schema changes, RTB still wants you to update the physical database with a Schema Update (it does not know that you are using the same DB across workspaces). Since the changes are already in the physical database from the previous workspace, you need to use the "skip" option on all tables.
* The CRCs will have changed for the table. A selective compile is used to recompile all programs that reference the changed table. The "other" workspaces will not know about the schema change until you bring over the logical changes, so your testers will get CRC errors when you do make a schema change.

Manually Managing Database Connections

The physical database connection specified in the physical database connection parameters is used in a standard Progress "CONNECT" statement. It is not used to "tie" the logical object to the physical object. As described in the "compiling" section, the Progress compile against the physical DB and the XREF against the logical objects in RTB are separate and isolated. You simply need to be connected to a physical database to allow a successful Progress compile and you need logical definitions within RTB to satisfy the XREF.

If you leave the physical name blank, RTB does not try to manage the connection to the physical DB when you select the workspace. This allows you to manage the connections manually. You can add menu choices to RTB to connect to specific databases after selecting the workspace. In V8.3B02 and higher, you can use the change workspace hook to manage the connection to the databases. This functionality is useful for things such as using schema sandboxes or only connecting to a desired subset of databases.