Database Extensions Best Practices

Presenter: Roger Sprik PSUG National Conference July, 2021

Released in PowerSchool 7.9

Key Documents on Community and PowerSource:

- Database Extensions Visual Walkthrough - ID: 77671
- Database Extensions Advanced User Guide for PowerSchool 9.x - ID: 74828
- Database Extensions Best Practices - ID: 71545
- Database Extension and Custom Field Migration Frequently Asked Questions for PowerSchool 8.x - ID: 72504
- Data Dictionaries Tables for PowerSchool 12.x - ID: 80621

Overview

- Database Extensions is the name for the new method for custom data entry points in PowerSchool.

- It will currently run side-by-side with the old way of doing custom data, but eventually will completely replace it. Migration of user-created custom fields is not yet required as of version 19.x. However, there are new areas that cannot access legacy custom fields, such as Enterprise Reporting or the API.

- **Before**:
    - Users go to System - Custom Fields/Screens
    - Only Student, Staff (teachers table), Course and Section fields were available to be created this way
    - Only text fields could be created
    - All data went into one giant custom field that had to be parsed
    - The only way to do one-to-many custom fields was to use virtual tables which is little known and very difficult to work with

- **After:**
    - Users go to System - Page and Data Management - Manage Database Extensions
    - Users create REAL tables and fields of various data types.
    - Migration options for legacy custom fields
    - Many, many more tables can be extended, users can create one-to-many tables and even independent tables.
    - Inherent grouping of fields for logical organization.
    - No more 999 limit!
    - All tables get some tracking fields: whocreated, whencreated, whomodified, whenmodified. It won't tell you what changed, but who and when.
    - Any extended table can be exported from, and imported into, with the Data Export and Import Managers.

The Old Way. *With custom fields, all custom data is stored in a single field in the core table. They are all "Text" types.*

Students Table	
Last_Name	Adair
First_Name	Brandon
DCID	2
ID	5
CUSTOM	Medical Notes: Allergic to peanuts contact_1_email: contact1@email.com contact_2_email: contact2@email.com contact_1_employer: Wal Mart contact_2_employer: Lanham Law Offices parent_notes: Brandon is only allowed to be picked up from school on Wednesdays by his father.

Database Extensions - The New Way. *Faster, real tables and fields, multiple data types.*

Students	*Core Table "Parent"*		**U_Def_Ext_Students**	*New extended table "Child"*
Last_Name	Adair			
First_Name	Brandon		*This table "relates" to the Students table*	
DCID (*Primary Key*)	2	<->	StudentsDCID (*Foreign Key*)	2
ID	5		Contact1_Email	contact1@email.com
			Contact2_Email	contact2@email.com
			Contact1_Employer	Wal Mart
			Contact2_Employer	Lanham Law Offices
			Medical_Notes	Allergic to Peanuts

Custom Field Migration

- **Core Custom Fields.** The core PowerSchool product has many legacy fields that were never "real" fields, but were created by the developers as custom fields. These are known as "Core Fields".
- **Activities** have traditionally been stored as custom fields.
- **SuccessNet** integration fields were also custom fields.
- **User Custom Fields** are the fields created by customers

- History of options to migrate core custom fields and user custom fields.
    - 7.9 First set of core custom fields can be migrated
    - 7.10 Second set of core fields can be migrated. User fields can be migrated one at a time. Activities and SuccessNet fields migrated automatically
    - 7.11 User custom fields can be migrated all at once (*but you should NOT do this!!*)

- Core Fields Migration (required as of 10.1) (*See Data Dictionary for complete list*)
    - 1st 59 student core custom fields are moved to a new table called StudentCoreFields
        - ACT_Composite/Date/English/Math/Reading/Science, SAT
        - Emerg_1/2/3_Ptype/Rel
        - Guardian, Guardian_LN, Guardian_FN, Guardian_MN
        - Mo/Father_Home_Phone, Mo/Father_Day_Phone, Mo/Father_Employer
        - PrimaryLanguage, SecondaryLanguage
        - PrevStudentID, Family_Rep, Area, Dentist_Name, etc
    - 2 teacher/user core custom fields are moved to a new table called UsersCoreFields
        - DOB
        - Gender
- Core Fields 2 Migration (required as of 10.1) (*See Data Dictionary for complete list*)
    - 52 more student custom fields are migrated to the StudentCoreFields table
        - Autosend fields
        - CRT_ fields
        - EC_ fields
        - IPT_ fields
        - a few others.
    - 1 course field is moved to the School_Course table
        - Alt_Course_Number
- User Custom Fields migration - Two (or three) ways:
    - **One-at-a-time**. You can migrate an existing custom field from the screen where you create a new extended field.
    - **All at once**. From System - Page and Data Management - Custom Field Data Migration.
    - **Start over.** Export all legacy data, delete the old custom fields, import into new database extension. Most tedious and legacy dependencies must be updated.
    - Which way is best? All at once is faster, but one at a time gives you the opportunity to organize your fields into logical groups and a chance to standardize your naming convention. Starting over is the cleanest, and will result in the cleanest field list.
- Migrated fields continue to operate the “old” way without changing any existing custom pages, exports, reports, etc. PowerSchool is programmed to act like migrated fields still exist in the “old” location, but there is a performance penalty. You will eventually want to use them with the “new” rules.
- New fields created AFTER migration must use the new naming and querying rules
- Please note the issues with migration of fields that contain more than 4000 characters.

Custom Field Migration is managed from:

System > Page and Data Management > Custom Field Data Migration

Organizational Concepts

It's important to understand the concept of an Extension Group vs an Extension table vs an Extension Field. A "Group" is a high level organization structure. It's a way to keep your tables and fields organized by function, such one group for managing Families, and another group for managing College Applications. You can also choose to have one group for ALL your custom fields. If you plan on sharing database extensions, you should consider multiple groups. You should choose your names for groups carefully -- your users will have to type them, so keep them intuitive and short.

Remember the Flow

Core Table > Extension Group(s) > Extension Table(s) > Extension Field(s)

Cautionary Notes:

- Plan and think about your users and customizations when naming them. Even though you now can delete extensions (if created via the UI), you should still proceed carefully.
- For each extension group's core table, you can only have one 1:1 extended table, but you can have multiple 1:many tables.
- When you install someone else's plugin and it contains extensions, they will be added to your system and the tables and fields cannot be removed, only abandoned.
- When you disable a plugin, it won't be served from CPM, but there is no way to tell in CPM if a custom page is associated with a plugin (unless you use Blue Steel CPM)
- ReportWorks will show extended fields, but must be restarted first.
- It's not obvious, but you can have more than one core table per extension group… just select the core table and then pick an existing extension group in Step 2.
- The Database Extension wizard suggests group names like U_Students_Extension and table names like U_DEF_EXT_STUDENTS --- you DO NOT have to use those names, you might want to use names that make sense for your application. DELETE them in the wizard FIRST if you are not going to use the suggested name.
- "Group" name vs table name in 1:1 extensions can be confusing. In some areas you have to refer to the group name, in others the table name. Some users advocate making the name of the group the same name as the table for this reason. i.e. U_VEHICLES

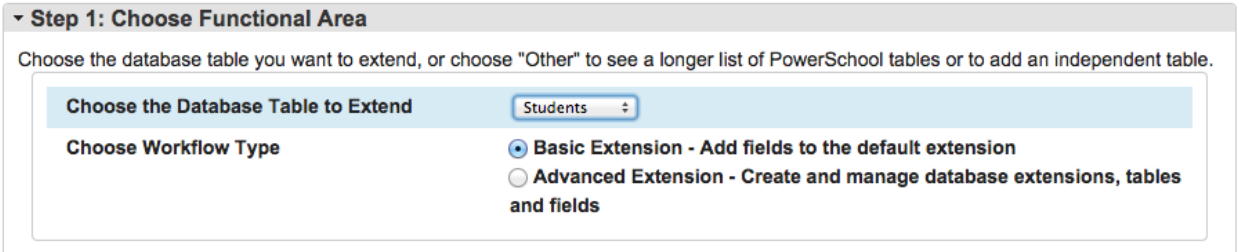
Creating Database Extensions

There are more complete instructions in the PowerSource documents, these are the basics.

System > Page and Data Management > Manage Database Extensions

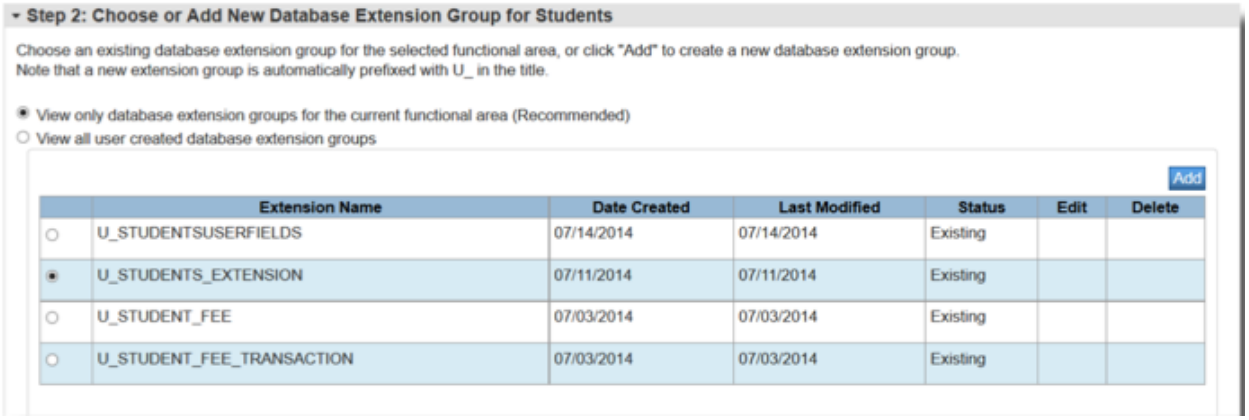
There are 4 steps to creating a database extension

Step 1 - Choose the core table that will be extended. You can choose Basic or Advanced Extension. The Basic option will automatically select default options for steps 2 and 3, bringing the user directly to step 4.



Step 2 - Choose or add a database extension group. The database extension group is simply a name given to a group of database extended tables. Many users choose to use a single extension group for all custom fields. Others choose to create different groups to compartmentalize data (i.e. demographic data and contact data).

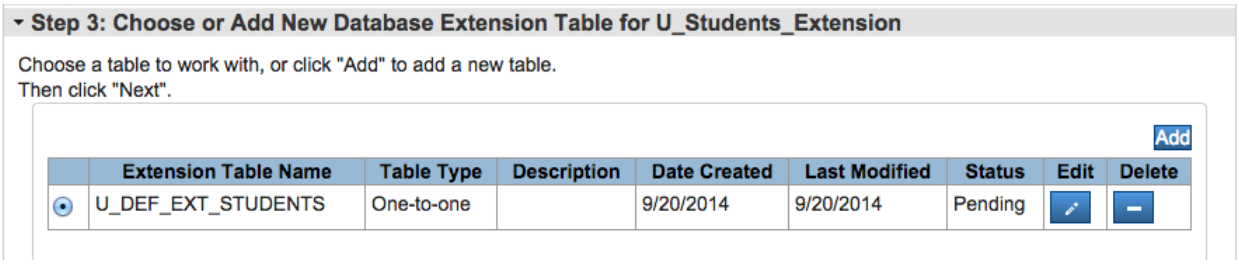
U_Students_Extension is the default extension group when extending the Students table. This is the option that will be used if the Basic Extension type is selected in step 1.



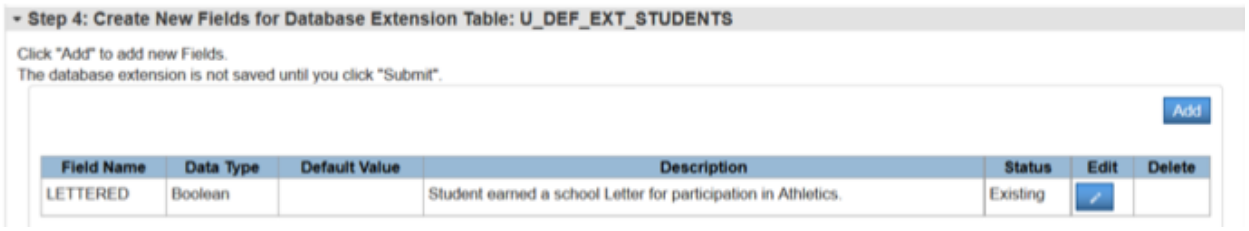
Step 3 - Choose or add a database table. Each extension group can contain multiple tables. There are two types of tables.

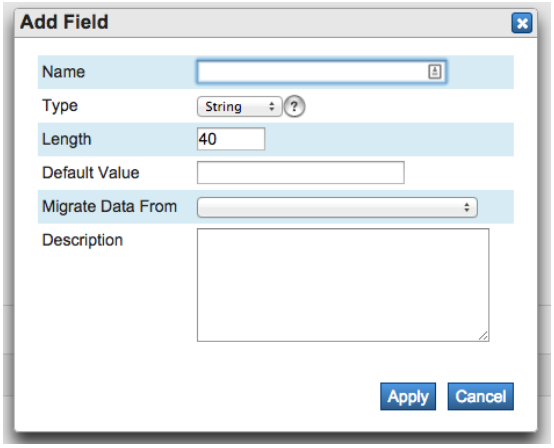
- **One-to-One tables** hold data that can be defined only once for a student. These are the tables that will hold new or migrated custom fields.
- **One-to-Many tables** can have multiple records for a single student. Examples of these types of tables in core PowerSchool include CC, StoredGrades, and Log

U_Def_Ext_Students is the default extension table when extending the Students table. This is the option that will be used if the Basic Extension type is selected in step 1.



Step 4 - Create new field for the selected table. Here the user can create new fields. When the Add button is clicked, the Add Field dialog will appear.





Field	Description
Name	Enter the field name as it will appear in PowerSchool.
Type	Select the filed data type from the pop-up menu: • **String**: Fixed-length character string. Strings in excess of 4000 characters will be truncated. • **Integer**: A number that can be written without a fractional or decimal component. • **Date**: A point-in-time value. • **Double**: An approximate representation of a decimal value. • **Boolean**: Data has two values (1=true and 0=false). • **CLOB**: (Character Large Object) data stored in a separate location referenced by the table. • **BLOB**: (Binary Large Object) collection of binary data stored as a single entity.
Length	Enter the maximum length of data that can be entered in the field.
Default Value	Enter the default value for the field.
Migrate Data From	Choose a legacy custom field in order to migrate data from the selected field to your database extension field. The default Type is set to String and the default Length is set to 4000 automatically. This option is unavailable if there are no existing legacy custom fields.
Description	Enter a brief description of the field.

Important notes about field creation

- Take time to consider names, data-types, and lengths carefully before submission.
- Leave the 'Migrate Data From' dropdown empty to create a new field.
- If migrating a new extended field will be created and all data from the custom field will be migrated into the newly-defined extended field. The data will also be kept in sync with the legacy field, preserving the functionality of any old reports or customizations that refer to the legacy field.
- Fields can be renamed when migrating. PowerSchool maps previous names to the new extended names to ensure that no custom pages or queries will break.

Using One-to-One Extensions

- Searching, exports, list students (wherever the PS application itself asks you for a “fieldname”)
  - ExtensionGroupName.Field_Name.
  - Example: U_Students_Extension.DistrictID
- Reports that use DATs (wherever you normally put ~(fieldname))
  - ~(ExtensionGroupName.Field_Name)
  - Example:~(U_Students_Extension.DistrictID)
- HTML pages
  - [PrimaryTable.ExtensionsGroupName]FieldName
  - Example: [Students.U_Students_Extension]DistrictID
- SQL Queries.
  - The extended database tables link on {CORETABLE}DCID

```
SELECT * FROM
CoreTable
LEFT JOIN ExtendedTable  ON CoreTable.DCID = ExtendedTable.StudentsDCID
```

- EXAMPLE:

```
SELECT
s.lastfirst, s.grade_level, s2.districtid

FROM
students s
LEFT JOIN U_DEF_EXT_STUDENTS s2 ON s.dcid = s2.studentsdcid

WHERE
s.enroll_status=0 AND s.grade_level=12 AND s.schoolid = 100
```

One-to-Many Extension Tables

A one-to-many extended table allows you to have multiple records that are tied back to a single parent record. For example, multiple college applications for a single student.

- To work with one-to-many tables you use the tlist_child tag. This tag will create a table, so use it outside of any other tables, but still within a standard type page that has a form tag with the usual submit button and associated hidden inputs.

  ~[tlist_child:<CoreTableName>.<ExtensionGroup>.<ExtensionTable>;displaycols:< List of Fields>;fieldNames:<List of Column Headers>;type:<FormatName>]

  Example:

  ~[tlist_child:Students.U_CollegeApp.U_Applications;displaycols:Institution,Request_Date,Status;fieldNames:Institution,Request Date,Status;type:html]

- SQL query example for a one-to-many table.

```
SELECT
        s.lastfirst,
        app.institution,
        app.request_date,
        app.status
FROM
        students s
        LEFT JOIN U_Applications app ON s.dcid = app.studentsdcid
WHERE
        s.id = 2
```

Independent (Standalone) Extension Tables

Creates a table that is not associated with any existing PowerSchool table. An example might be a list of college institutions.

- To work with independent tables use the tlist_standalone tag. It works much like the tlist_child tag.
  ~[tlist_standalone:<ExtensionGroup>.<ExtensionTable>;displaycols:<List of Fields>;fieldNames:<List of Column Headers>;type:<FormatName>]

Example:
~[tlist_standalone:U_CollegeApp.U_Institutions;displaycols:Institution_Name,Phone,URL;fieldNames:Institution Name,Phone Number,Web Address;type:html]

- Example SQL Query for Independent table:
  SELECT * FROM U_Institutions

Even more advanced?

- Refer to the Advanced User Guide for Database Extensions for special formatting of tlist_child and tlist_standalone tags.
- Consider using the DirectTable.Select tag for an advanced way to use one-to-many and standalone tables. The "Database Extensions Advanced User Guide…" has a good College Application example implementing this alternate method to tlist_child and tlist_standalone.

Custom Resource

Erich Schaitel from Marcia Brenner Associates has a great "Customization Reference" plugin that documents how to customize in PowerSchool, including Database Extensions and using the DirectTable.Select tag. Get it at:
https://mba-link.com/powerschool-plugins/#free-plugins

Planning and Preparing for Migrating your user-created custom fields

Planning, planning planning! That's the key. Write a data dictionary for your school's custom fields. Use PowerSchool's data dictionary format as a template. It is HIGHLY recommended NOT to use the "all-at-once" approach to migrating your fields. That method will make all your fields a text type with a length of 4,000, place them all in the same group, and give you no choice over the name of your group or extended table names. Use this as an opportunity to do some "spring cleaning", retiring custom fields you no longer use and reviewing your naming conventions.

You will also want to make sure your data is ready for migration.

- Do you have any fields with more than 4,000 characters?
- Are you migrating a text field to a date field?
- Are you sure the date formats are all correct?

You should seriously consider the customization by Peter Nethercott - "**Custom Legacy Field Inspector - Extension Migration**".
https://support.powerschool.com/exchange/view.action?download.id=746

It can identify what legacy fields you have AND tell you:

- how and where they are used
- how much they are used
- what types of values are stored in them
- whether they have been migrated or not
- if already migrated, the name of the extension to which they have been migrated

It can help with the cleanup before migration, including the addition of links directly to the object report, AutoComm/AutoSend, Custom Screen definition, etc., that are listed on the resulting reports