WORKING WITH EXTERNAL DATABASES

DireXions 2024

AGENDA

PxPlus External Database Support

Prefix File

Link File

Using External Databases

External Database Import

External Database Export

How to Convert to External Databases

PXPLUS EXTERNAL DATABASE SUPPORT

PXPLUS EXTERNAL DATABASE SUPPORT

- While most aspects of a business application can be served within the PxPlus family of products, today's end users are often required to work with data that resides in completely different software worlds
  - Businesses may need to integrate popular "off-the-shelf" software with their legacy systems, applications and databases
- PxPlus can work directly with data in a number of **External Databases,** over networks and on completely different operating systems
  - Any external database with an ODBC driver via [ODB]
  - MySQL/MariaDB via [MYSQL]
    - Requires MySQL/MariaDB C Connector: libmysql.dll
  - Microsoft SQL Server via [ADO]
  - Oracle Server via [OCI]
    - Requires Oracle instant client: oci.dll
  - IBM DB2 via [DB2]
    - Requires DB2 CLI client: db2cli.dll

PXPLUS EXTERNAL DATABASE SUPPORT

- An External Database table is accessed via an **OPEN** directive just like a PxPlus native file
  - Use the **[xxx]** prefix depending on the type of database
  - The database connection information and table name are given either after the **[xxx]** prefix or via **OPT=**
    - Check the documentation of the specific database prefix you want to use for the exact syntax

```
open(hfn,iol=*,OPT="SERVER=192.168.1.114;PORT=3306;USER=xxxx;PSWD=xxxxxxx")"[MYSQL]test_db;invoice_header"
```

- For Read-only data that is not too large, dramatically improve performance with **OPEN LOAD** directive
  - Locally caches the whole table in memory
  - Controlled system wide via the **'CL'=value** system parameter
    - All tables with a record count below value will be cached, 0 will disable **OPEN LOAD** caching, default is 1000
  - Individually controlled via **OPEN LOAD** directive when **OPT=** clause of **CACHE=value** is included, overriding **'CL'**
    - A value of *yes* means to always cache, *no* means to never cache, *nnn* means to cache if record count is less than *nnn*

```
open load(hfn,iol=*,OPT="SERVER=192.168.1.114;PORT=3306;USER=xxxx;PSWD=xxxx;cache=yes")"[MYSQL]test_db;invoice_header"
```

PREFIX FILE

PREFIX FILE

- Can define a new prefix file entry with the database connection information
  - Code can be kept cleaner or be switched from native file to a database without changing code
  - The prefix file is a PxPlus native variable length keyed file
  - Prefix file entry/record
    - The key is the name you want to access your database table by
    - The first field will contain the database prefix and the DSN/Table declaration
    - The second field will contain the database connection options usually in the **OPT=**
    - The third field may be specified that contains the IOList to use when opening the file with an **IOL=*** option

```
KEYED "PFXFILE",127
OPEN (1) "PFXFILE"
WRITE(1,KEY="my_table")"[MYSQL]test_db;my_table;SERVER=192.168.1.114;PORT=3306;USER=xxxx;PSWD=xxxxxxx",
"KEY=field1;REC=field1:10,field2:8.2,field3:40,field4:8,field5:2,field6:20",
"field1$,field2,field3$,field4$,field5,field6$"
CLOSE (1)
```

PREFIX FILE

- Use the **PREFIX FILE** directive to start using a prefix file
  - Once setup any open using the name key will actually open the database connection defined in the prefix entry

```
PREFIX FILE "PFXFILE"
OPEN (1,iol=*) "my_table"
```

LINK FILE

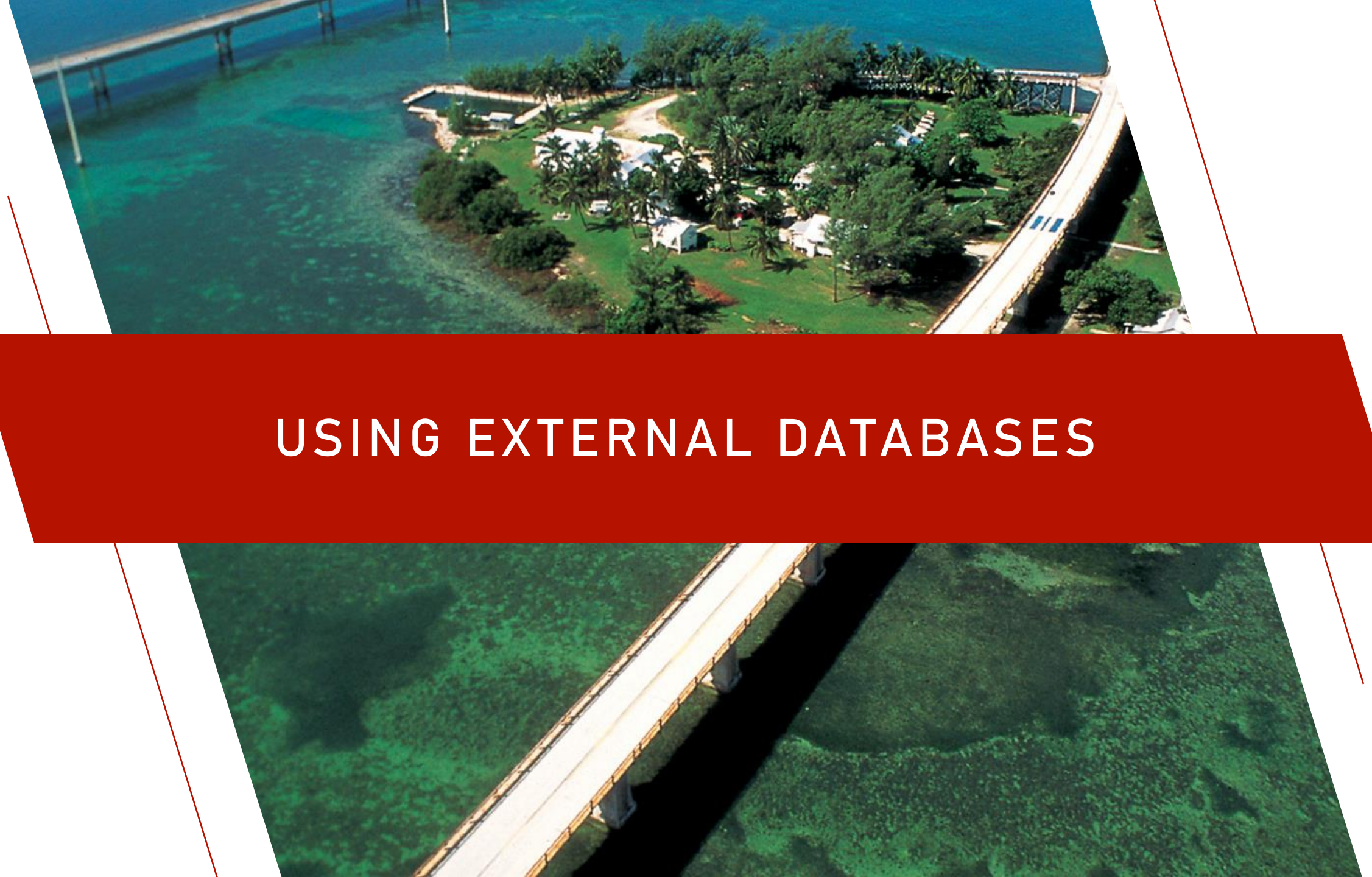
LINK FILE

- Can define a link file with the database connection information
  - Code can be kept cleaner or be switched from native file to a database without changing code
  - An external database link file is a [Pvxdev] **Link File** that points to a device driver, ***dev/extdb**
  - Link File Format:
    - Line 1: Typical line for a [Pvxdev] link file, pointing to the device driver, *dev/extdb (256 characters)
    - Line 2: Database Type
    - Line 3: Database Name
    - Line 4: Table Name
    - Line 5 (and higher): Options
  - Any line in the link file (after the first line) that starts with an **=** (equals sign) will be evaluated
    - This can be used to avoid plain text password in the link file
      - ="USER="+%adoUser$
      - ="PSWD="+%adoPswd$
  - The **IOL=** and **OPT=** on the **OPEN** of the link file will be used
    - Any OPT= on the OPEN of the link file overrides the OPT= defined in the link file

LINK FILE

```
[Pvxdev].                                    extdb
ADO
ServerName
TableName
DB=databaseName
NONULLS=YES
Connect='Provider=SQLOLEDB;'
EXTROPT=(UPDLOCK)
DATEFMT=YYYYMMDD
KEY=fieldOne,*NAME:KeyOne
KEY=fieldTwo,fieldThree,*NAME:KeyTwo
REC=fieldOne:12,fieldTwo:40,fieldThree:6.2,fieldFour:6.0
```

- Suppose that you created a link file called "ADOProduct" that defined an ADO connection to the Product table
  - OPEN (chan,IOL=*)"ADOProduct"
- This will then make a connection to the external database as defined in the link file
- It would be as if "ADOProduct" was a PxPlus data file to your program even though it is an external database table



USING EXTERNAL DATABASES

USING EXTERNAL DATABASES

- Existing code using native PxPlus files will work without changes
- Queries, Reports, File Maintenance, and Webster+ equivalents will work without changes
- **READ**, **WRITE**, **INSERT**, **UPDATE** and **REMOVE** directives will generate the SQL to work with the database automatically
  - Can generate better optimized SQL if the keys are defined to match the tables index fields

PxPlus Code		SQL Code
open (1)"[ODB]MYDB;CUSTOMER" read record (1)R$	→	SELECT * FROM CUSTOMER
open (1)"[ODB]MYDB;CUSTOMER;KEY=CST_ID" read record (1,key= "00420000")R$	→	SELECT * FROM CUSTOMER WHERE CST_ID = "00420000"

USING EXTERNAL DATABASES

- **SELECT**, which allows you to use SQL-like syntax in your programs to access native files, also will convert into real SQL to work with the database automatically
  - A **WHERE** clause in the **SELECT** directive can optimize the SQL generated
    - Only if comparing field variable against literal

```
SELECT * FROM "[ODB]MYDB;CUSTOMER"
WHERE  State$ = "ON"
```

```
SELECT * FROM CUSTOMER WHERE State = 'ON'
```

    - Use **STATIC** clause to optimize if comparing field variable to a variable

```
SELECT * FROM "[ODB]MYDB;CUSTOMER"
STATIC WHERE  State$ = curState$
```

USING EXTERNAL DATABASES

- You can enable a debug mode where the SQL commands generated will be displayed via a msgbox
  - SET_PARAM '!Q'
- Directly execute SQL on the database
  - Use a **key="!<SQL>"** on a **READ** directive
  - Read Record also supported
  - Table name from connection ignored

```
open (1)"[ODB]MYDB;some_table"
sqlcmd$="SELECT first_name FROM students WHERE student_id IN (SELECT student_id FROM grades WHERE grade = 'A')"
read record (1, key="!"+sqlcmd$) result$
```

USING EXTERNAL DATABASES

- Open a direct connection to the database by **not** specifying a table name
  - If connection kept open can be used to avoid needing login credentials on each database connection
  - Can query the following with **key=**

KEY=	Action	SQL Function
"?"	Returns the list of table, catalog, or schema names, and table types	SQLTables()
"*xxxx"	Returns the list of column names in table *xxxx*	SQLColumns()
"**xxxx"	Returns a list of statistics about table *xxxx* and the indexes associated with the table	SQLStatistics()

```
open (chan)"[ODB]MYDB;"
read (chan, key="?“) tableCatalog$,tableSchema$,TableName$,tableType$,Remarks$
while 1
read (chan, err=*break) tableCatalog$,tableSchema$,TableName$,tableType$,Remarks$
wend
```

USING EXTERNAL DATABASES

- Open a direct connection to the database by **not** specifying a table name
  - **WRITE RECORD** directive allows you to execute SQL directly while the **READ RECORD** directive returns the results

```
open (chan)"[ODB]MYDB"
sqlcmd$="SELECT first_name FROM students WHERE student_id IN (SELECT student_id FROM grades WHERE grade = 'A')"
write record (chan) sqlcmd$
read record (chan) result$
```

  - Same **key="!<SQL>"** on a **READ** directive works here too

EXTERNAL DATABASE IMPORT

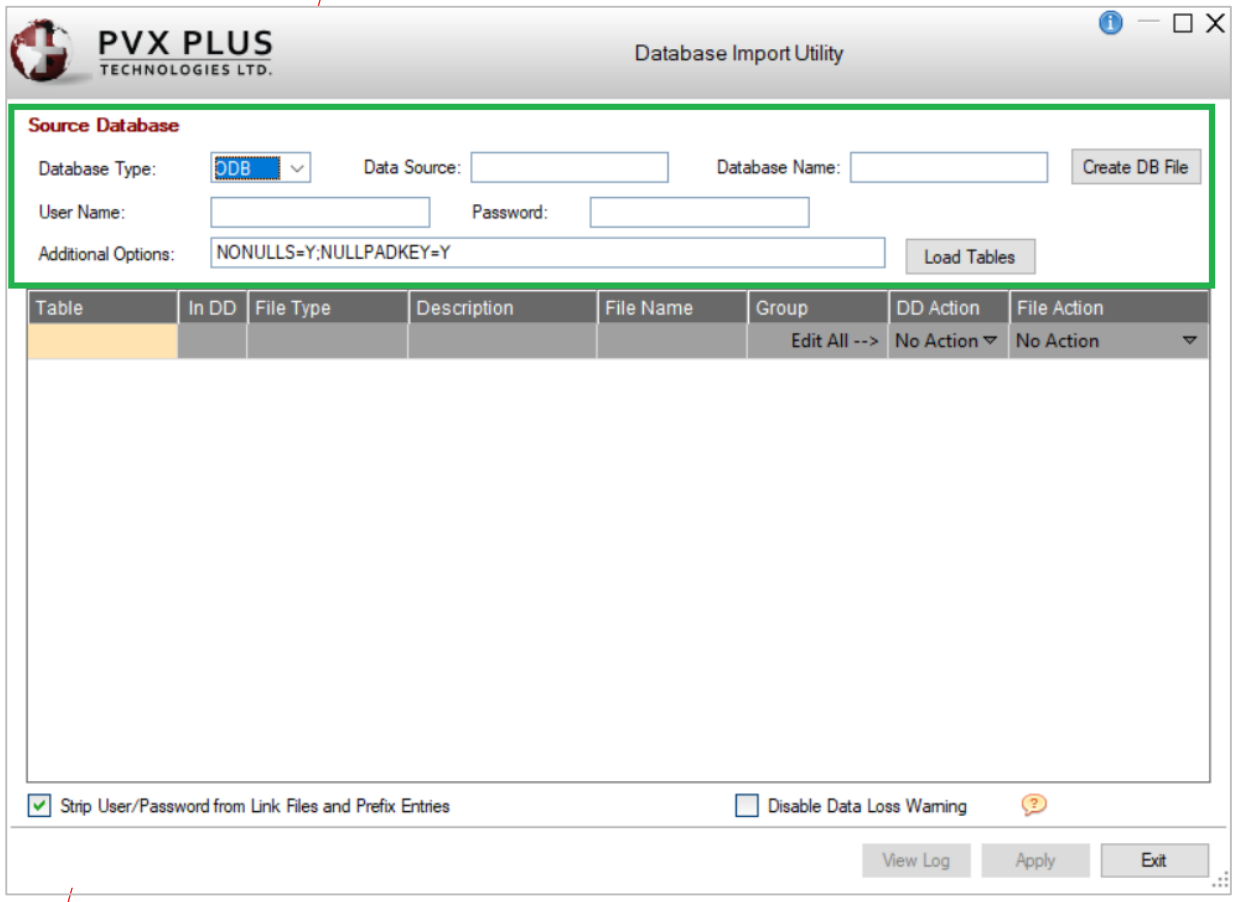
EXTERNAL DATABASE IMPORT

- **Database Import Utility** allows you to define a connection to an external database and then selectively import tables into the PxPlus data dictionary
  - Create **Link Files** that point to tables in the database
  - Create Link Files that point to the database itself
  - Add entries to the **Prefix File** that point to tables in the database
  - Add entries to the Prefix File that point to the database itself
  - Create and load data from database tables to native PxPlus data files
- To invoke this utility, use one of the following methods:

Location	Method
From **Data Dictionary Maintenance**	Click the **Import** toolbar button in the **Database** section of the toolbar
From **Data Dictionary Maintenance**	From the menu bar, select **File** > **Database Import Utility**
From the PxPlus Command line	Enter: **RUN "*dict/impdb"**

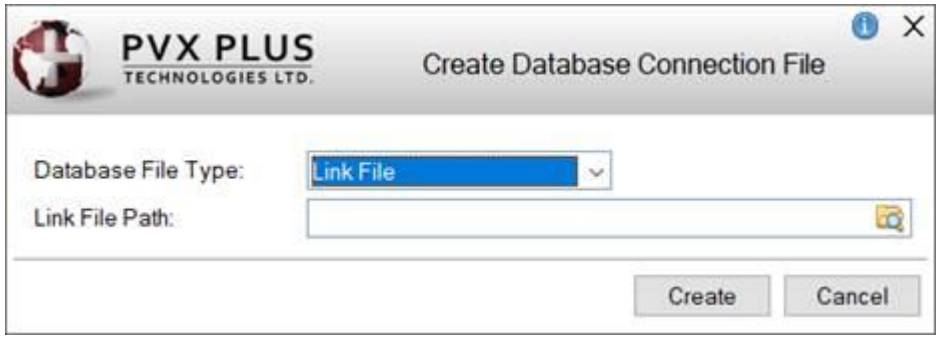
EXTERNAL DATABASE IMPORT

- To connect to a database to import from, you must input the database connection information
  - The utility remembers database connection information after successfully connecting to the database
    - Multiple database connections can be remembered, one connection for each database type
    - The connection information is remembered on a "per project" basis



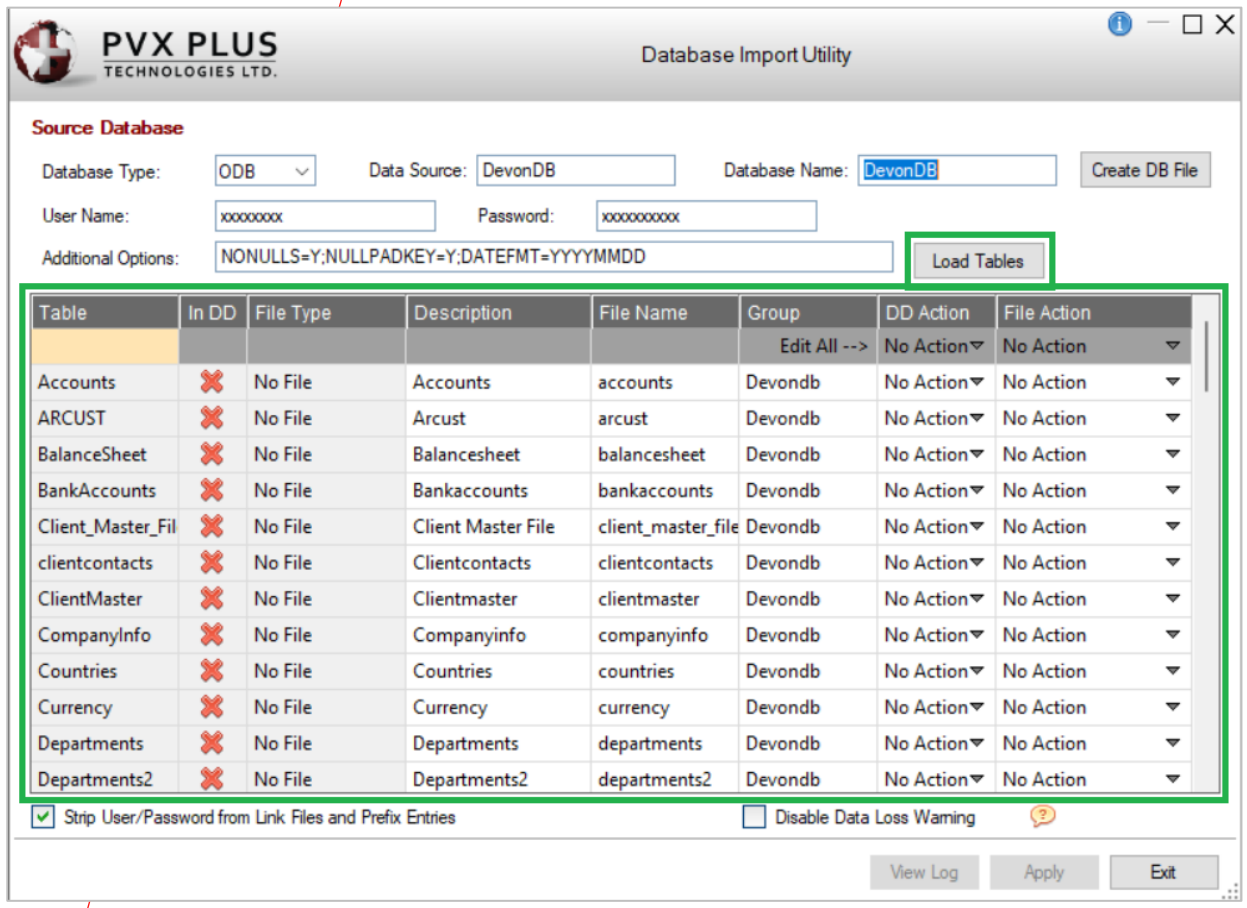
EXTERNAL DATABASE IMPORT

- **Create DB File** button allows you to create either a Link File or a Prefix File entry that points to the database itself
  - This may be useful so that you can do a global OPEN to the database, providing the UserID and Password to avoid needing the UserID and Password on every Link/Prefix File entry
  - It can also be used to get a list of tables from the database or if your program needs to make direct queries against the database



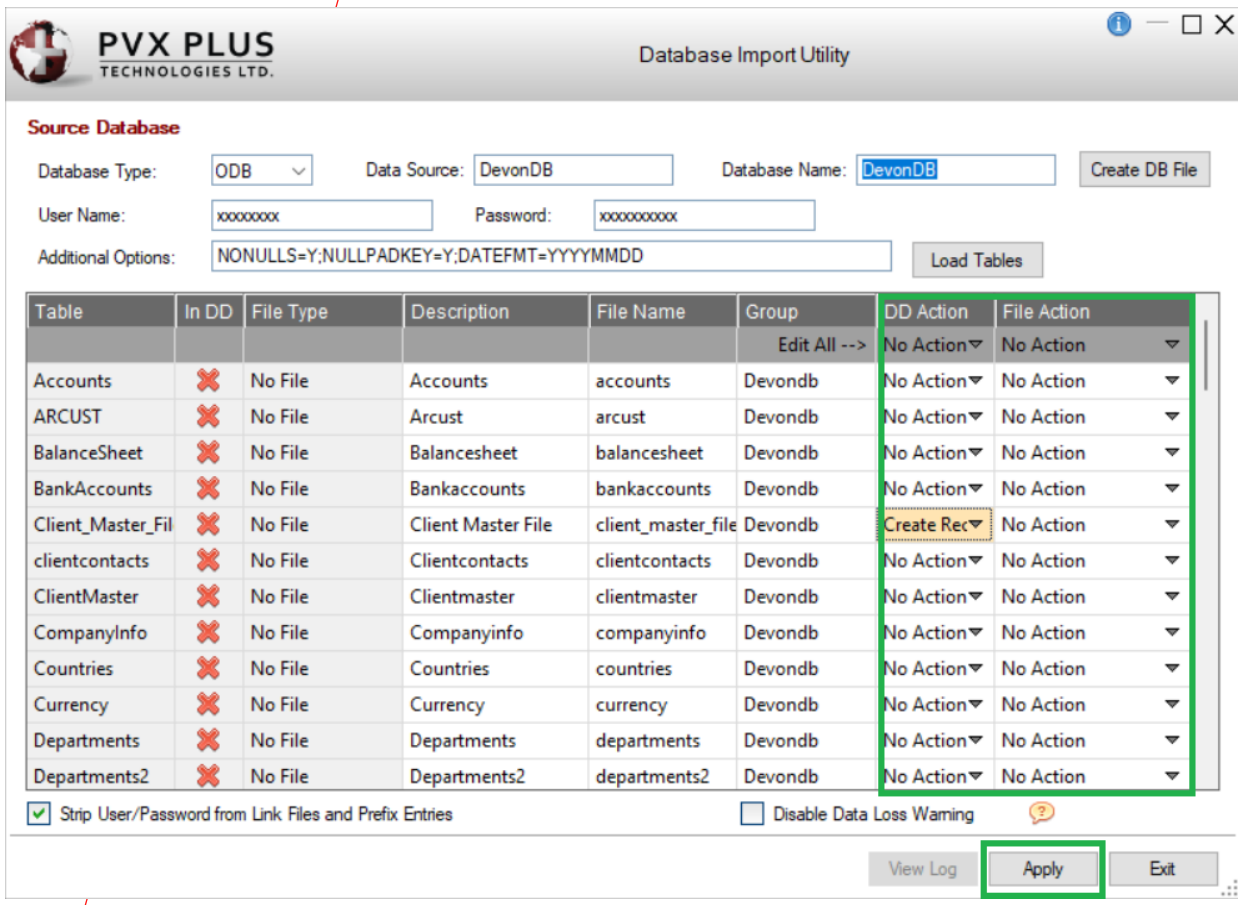
EXTERNAL DATABASE IMPORT

- **Load Tables** button - Lists the tables in the database, sorted in alphabetical order by default
  - **In DD** column - A check mark indicates which table definitions are currently in the data dictionary
  - **File Type** column - Tells you what type of file is in the data dictionary; i.e. No File, Native, ODB Link, etc.
  - **Description, File Name** and **Group** columns - Used to specify what the values should be on import and what they are for the imported table
    - If not in the data dictionary, will auto fill with table name from database



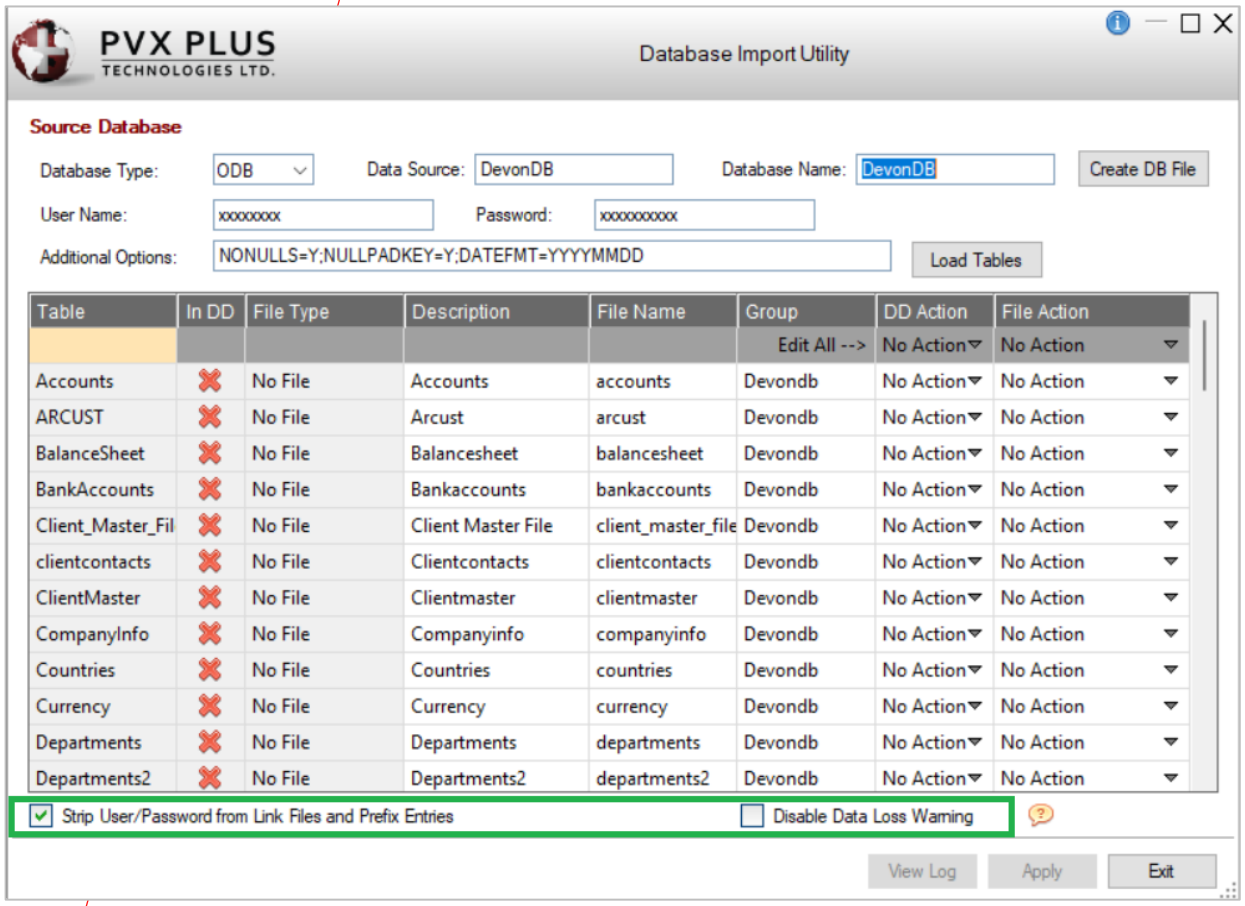
EXTERNAL DATABASE IMPORT

- **Actions** can be selected individually or for all tables
  - Use **Edit All** (first row) to select action for all tables
  - Use table row to select action for individual table
- **DD Action** - Can be used to:
  - Create data dictionary records
  - Merge data dictionary records
  - Replace data dictionary records
- **File Action** - Can be used to:
  - Create native PxPlus data files
  - Create database Link Files
  - Add database table entries to the Prefix File
- Actions are performed when the **Apply** button is clicked



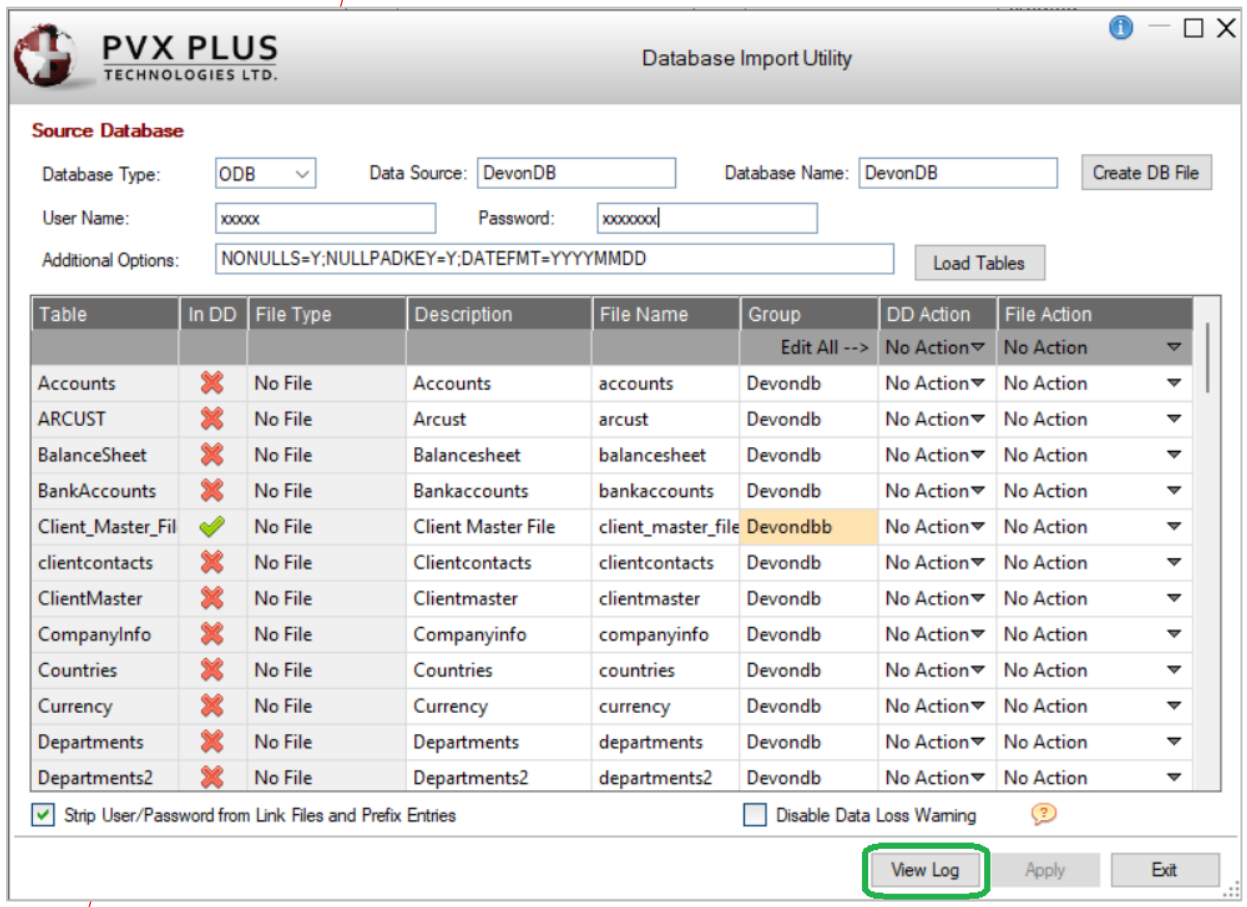
EXTERNAL DATABASE IMPORT

- Check box option to control whether User/Password are stripped from Link Files and Prefix File entries
  - **On** by default to prevent the User and Password information from being saved in plain text
- Check box option to control whether the data loss warning message is displayed
  - Disable warning for actions that could possibly overwrite data
    - Creating/Replacing a table
    - Loading a table
    - Creating a Prefix File entry
    - Creating a Link File
  - **Off** by default to help prevent accidental data loss



EXTERNAL DATABASE IMPORT

- **View Log** button
  - Available only when changes have been applied
  - Displays a log of applied changes as a PDF
  - Can print or save the log PDF, if desired



EXTERNAL DATABASE IMPORT

Importing external database tables programmatically via the *dict/impdb program

- The individual **Import** actions can be invoked programmatically
  - **Create/Merge/Replace a Database table into the Data Dictionary**

```
Call "*dict/impdb;CreateDDRecord", db_prefix$, connect$, db_name$, db_user$, db_pass$, db_options$, tableName$, description$, fileName$, groupName$, replace, result$
```

  - **Add to Prefix File Entry for the External Database table**

```
Call "*dict/impdb;AddToPfxFile", db_prefix$, connect$, db_name$, db_user$, db_pass$, db_options$, tableName$, fileName$, stripPswd, noDataLossWarning, result$
```

  - **Create Link File for the External Database table**

```
Call "*dict/impdb;CreateLinkFile", db_prefix$, connect$, db_name$, db_user$, db_pass$, db_options$, tableName$, fileName$, stripPswd, noDataLossWarning, result$
```

  - **Create and Load Physical File from the External Database table**

```
Call "*dict/impdb;CreateAndLoadFile", db_prefix$, connect$, db_name$, db_user$, db_pass$, db_options$, tableName$, fileName$, noDataLossWarning, result$
```

EXTERNAL DATABASE EXPORT

EXTERNAL DATABASE EXPORT

- PxPlus has two utilities to export a table to an external database
  - **Database Export Utility** - To export one table at a time
  - **Bulk Database Export Utility** - To export multiple tables all at the same time
- Utilities are similar in use and function so we will only cover the Bulk Database Export Utility
- The **Bulk Database Export Utility** allows you to define a connection to an external database and then selectively export tables from the PxPlus data dictionary
  - Create **Link Files** that point to tables in the database
  - Create Link Files that point to the database itself
  - Add entries to the **Prefix File** that point to tables in the database
  - Add entries to the Prefix File that point to the database itself
  - Validate native PxPlus data files
  - Load data from native PxPlus data files to database tables

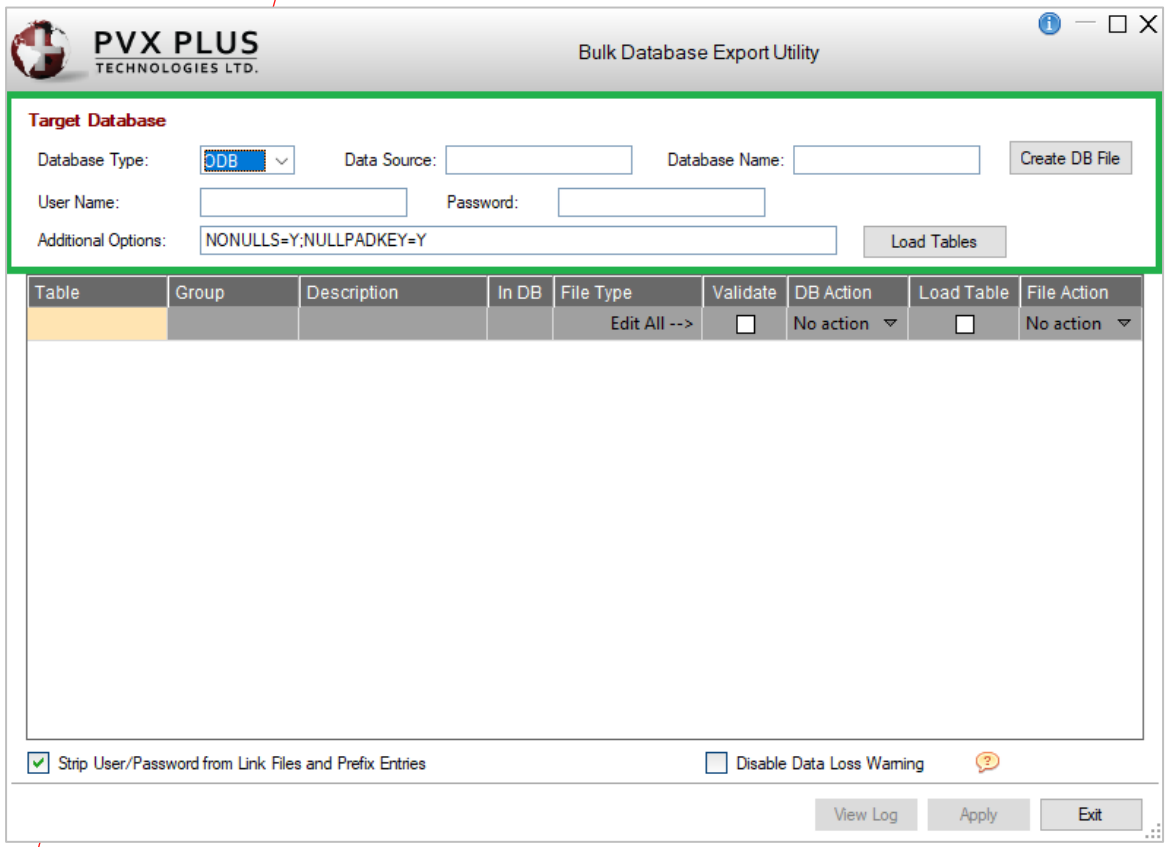
EXTERNAL DATABASE EXPORT

- To invoke the **Bulk Database Export Utility,** use one of the following methods:

Location	Method
From **Database Export Utility**	Click the **Bulk Export** button
From **Data Dictionary Maintenance**	Click the **Export** button in the **Database** section of the toolbar if no table with a native PxPlus data file is currently selected
From **Data Dictionary Maintenance**	From the menu bar, select **File** > **Database Export Utility** if no table with a native PxPlus data file is currently selected
From the PxPlus Command line	Enter: **RUN "*dict/expdde"**

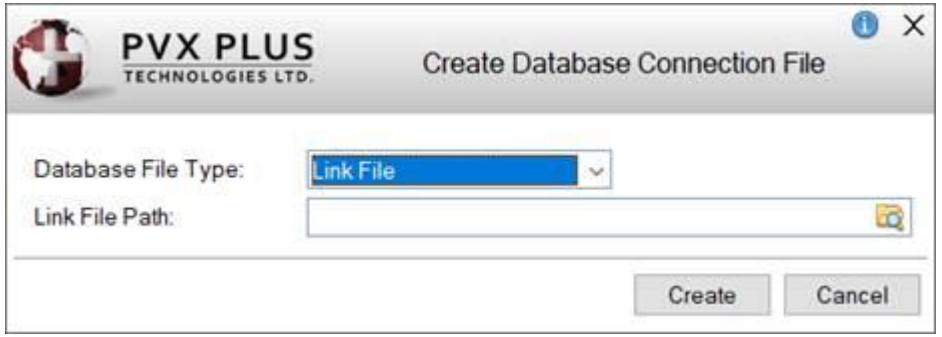
EXTERNAL DATABASE EXPORT

- To connect to a database to export to, you must input the database connection information
  - The utility remembers database connection information after successfully connecting to the database
    - Multiple database connections can be remembered, one connection for each database type
    - The connection information is remembered on a "per project" basis



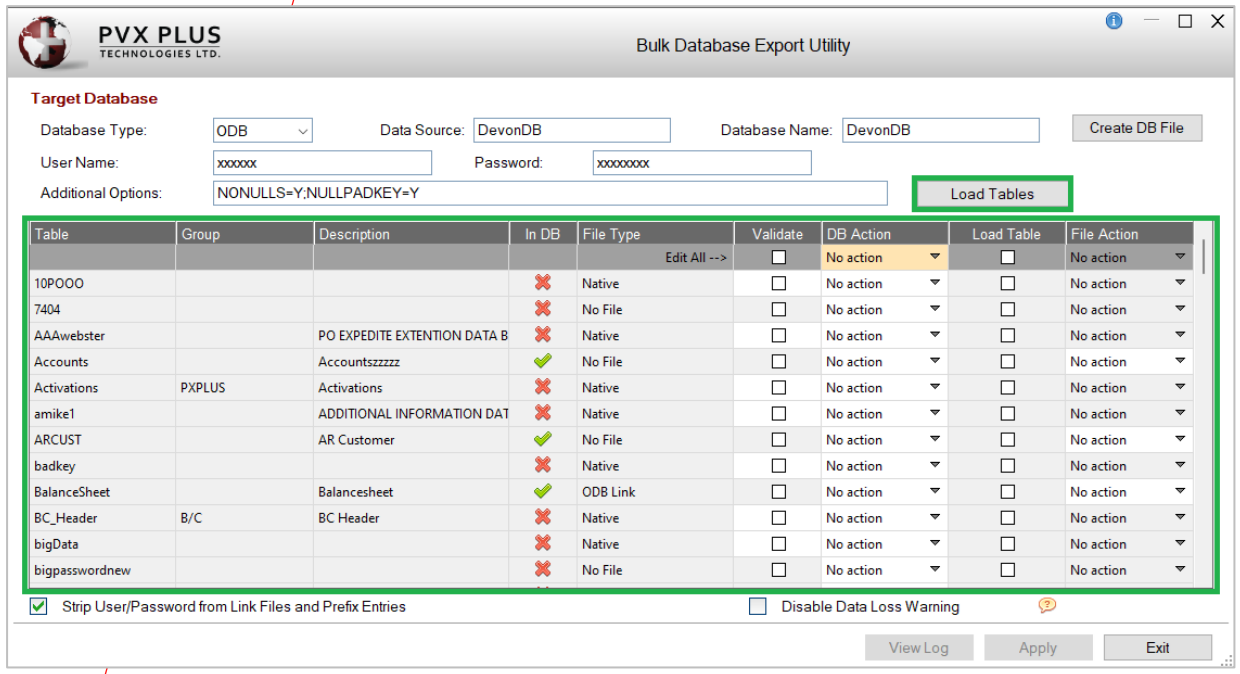
EXTERNAL DATABASE EXPORT

- **Create DB File** button allows you to create either a Link File or a Prefix File entry that points to the database itself
  - This may be useful so that you can do a global OPEN to the database, providing the UserID and Password to avoid needing the UserID and Password on every Link/Prefix File entry
  - It can also be used to get a list of tables from the database or if your program needs to make direct queries against the database



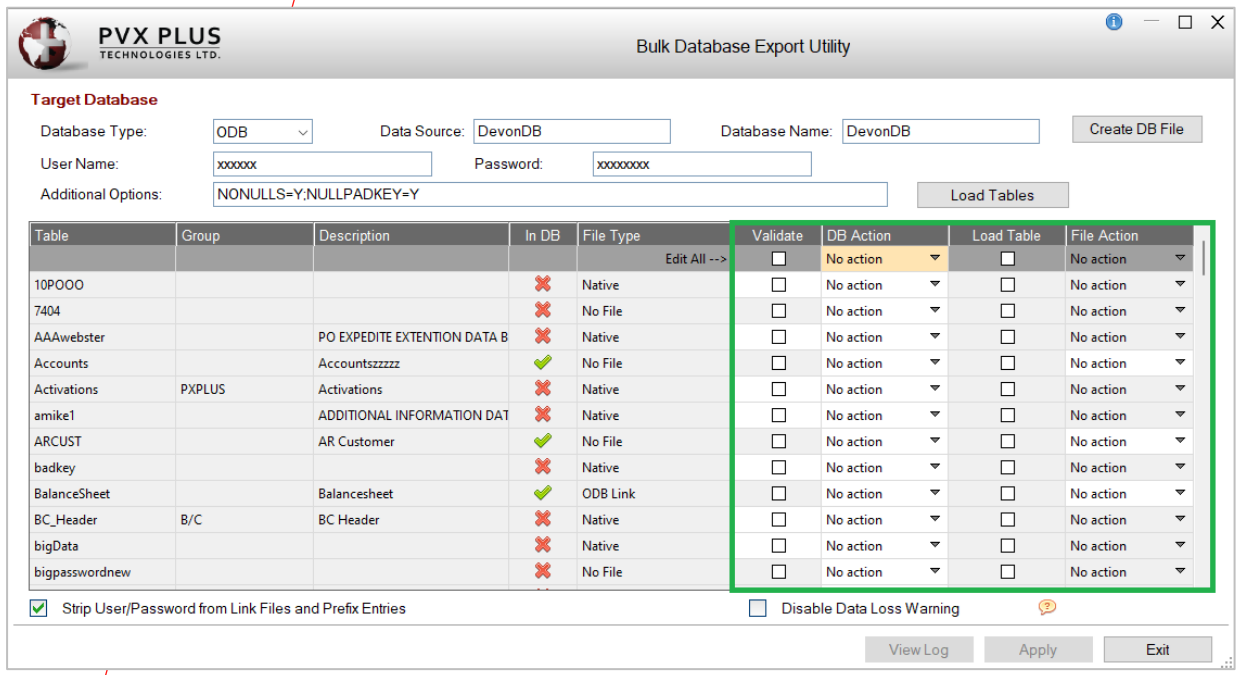
EXTERNAL DATABASE EXPORT

- **Load Tables** button - Lists the tables in the data dictionary, sorted in alphabetical order by default
  - **In DB** column - A check mark indicates which table definitions are currently in the database
  - **Group** and **Description** columns - Describe the table
  - **File Type** column – Tells you what type of file is in the data dictionary; i.e. No File, Native, ODB Link, etc.



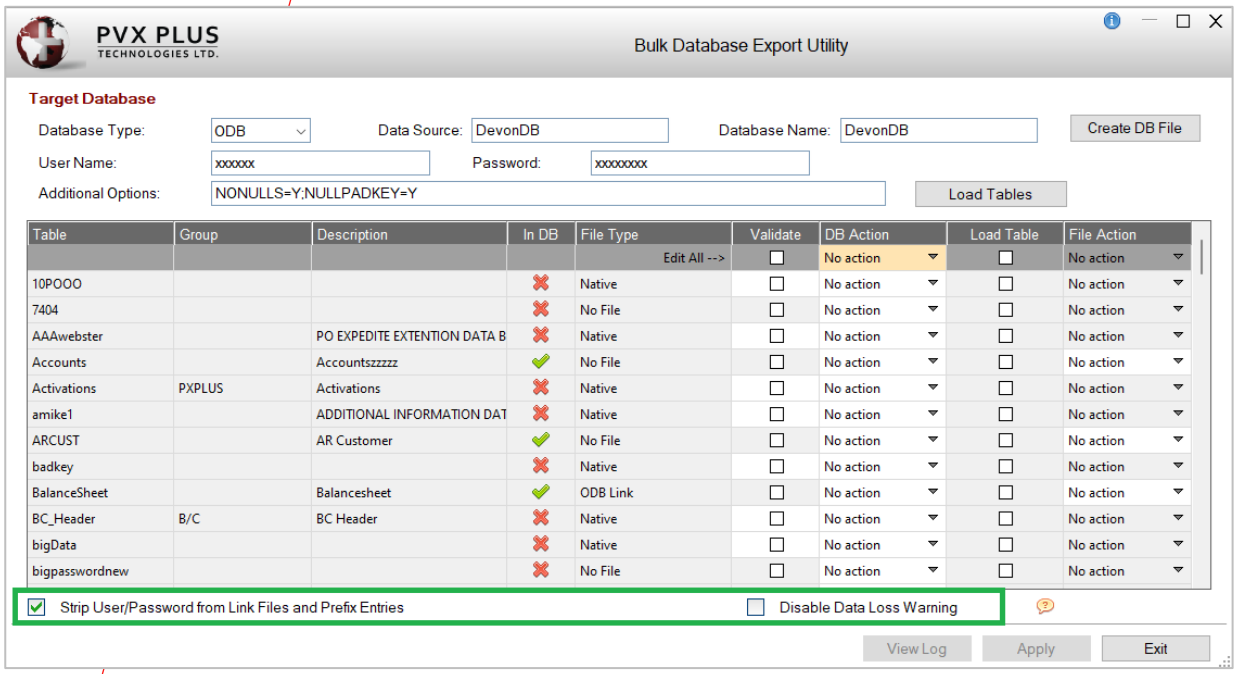
EXTERNAL DATABASE EXPORT

- **Validate** - Will have PxPlus check that the contents of the file match the elements definition
- **DB Action** - Can be used to:
  - Create table in external database
  - Merge table in external database
  - Replace table in external database
- **Load Table** - Will populate the database table with the data in the native PxPlus data file
- **File Action** - Can be used to:
  - Create database link files
  - Add database table entries to the prefix file
- Actions are performed when the **Apply** button is clicked



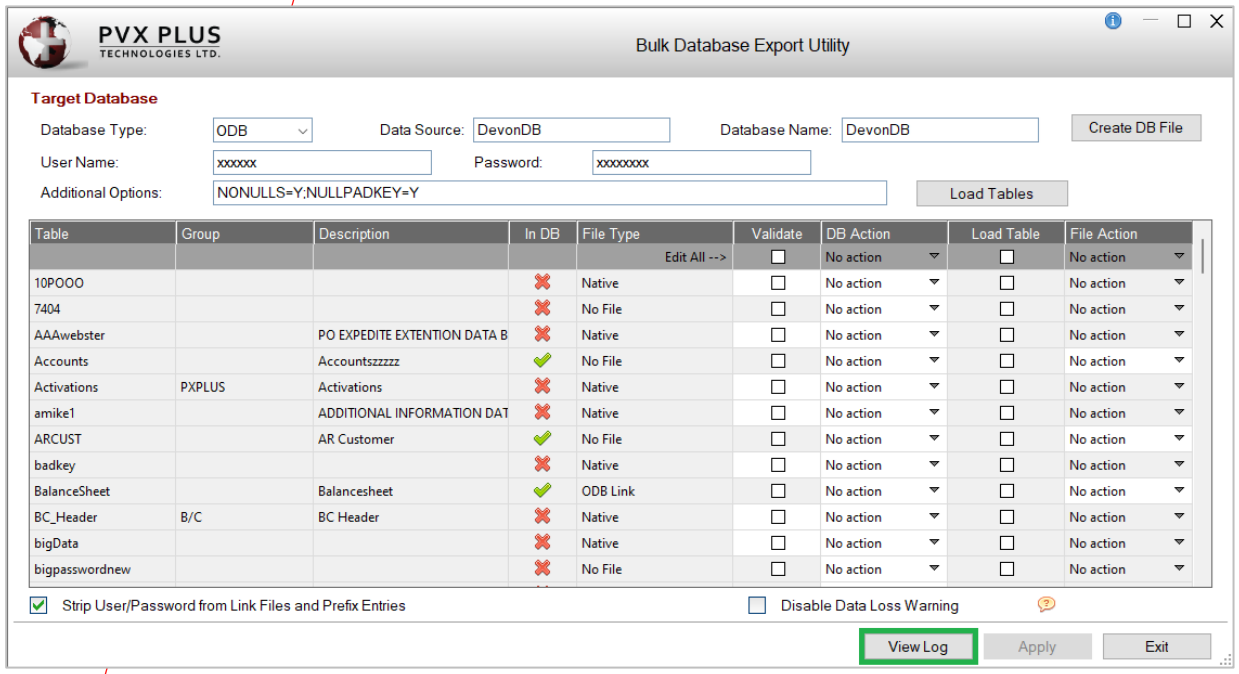
EXTERNAL DATABASE EXPORT

- Check box option to control whether User/Password are stripped from Link Files and Prefix File entries
  - **On** by default to prevent the User and Password information being saved in plain text
- Check box option to control whether the data loss warning message is displayed
  - Disable warning for actions that could possibly overwrite data
    - Creating/Replacing a table
    - Loading a table
    - Creating a Prefix File entry
    - Creating a Link File
  - **Off** by default to help prevent accidental data loss



EXTERNAL DATABASE EXPORT

- **View Log** button
  - Available only when changes have been applied
  - Displays a log of applied changes as a PDF
  - Can print or save the log PDF, if desired



EXTERNAL DATABASE EXPORT

Exporting tables to an external database programmatically via the *dict/sqlmake program

- **Create a table, Load a table and optionally Create the Prefix File entry and Link File**

```
Call "*dict/sqlmake;Convert",fileName$, db_type$, connect$, db_name$, db_table$, db_altname$, db_user$, db_pass$, db_options$,pfx_file$,link_file$,noOverwrite,noStripPrompt
```

- **Validate a PxPlus Native data file**

```
Call "*dict/sqlmake;Validate", fileName$, result$
```

- **Create/Merge/Replace a table in the Database from Data Dictionary table**

```
Call "*dict/sqlmake;CreateTable", fileName$, ddf_table$, db_prefix$, connect$, db_name$, db_table$, db_user$, db_pass$, db_options$, replace, result$
```

- **Load an External Database table with data from a PxPlus Native data file**

```
Call "*dict/sqlmake;LoadTable", fileName$, db_prefix$, connect$, db_name$, db_table$, db_user$, db_pass$, db_options$, overwrite
```

- **Add to Prefix File entry for the External Database table**

```
Call "*dict/sqlmake;PrefixFile", fileName$, db_prefix$, connect$, db_name$, db_table$, db_altname$, db_user$, db_pass$, db_options$, pfx_file$, stripPswd
```

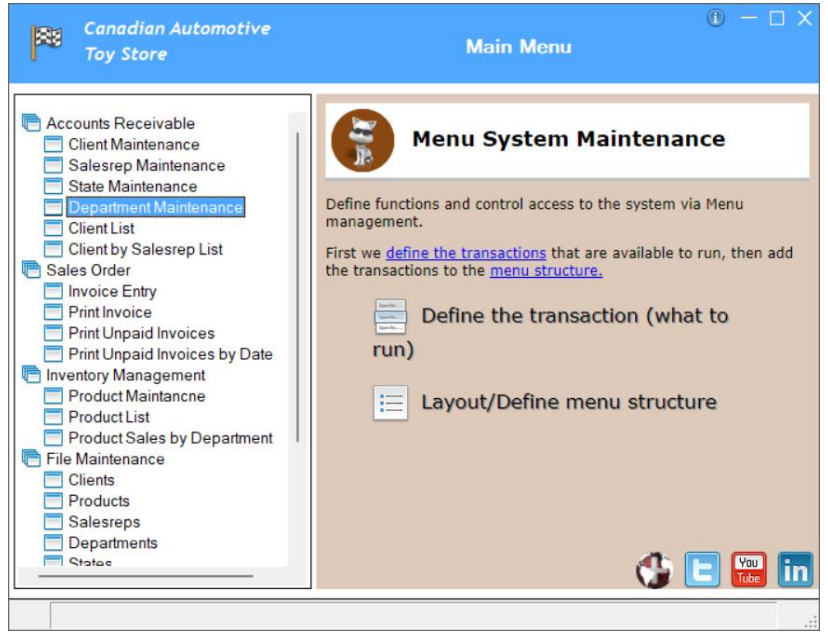
- **Create Link File for the External Database table**

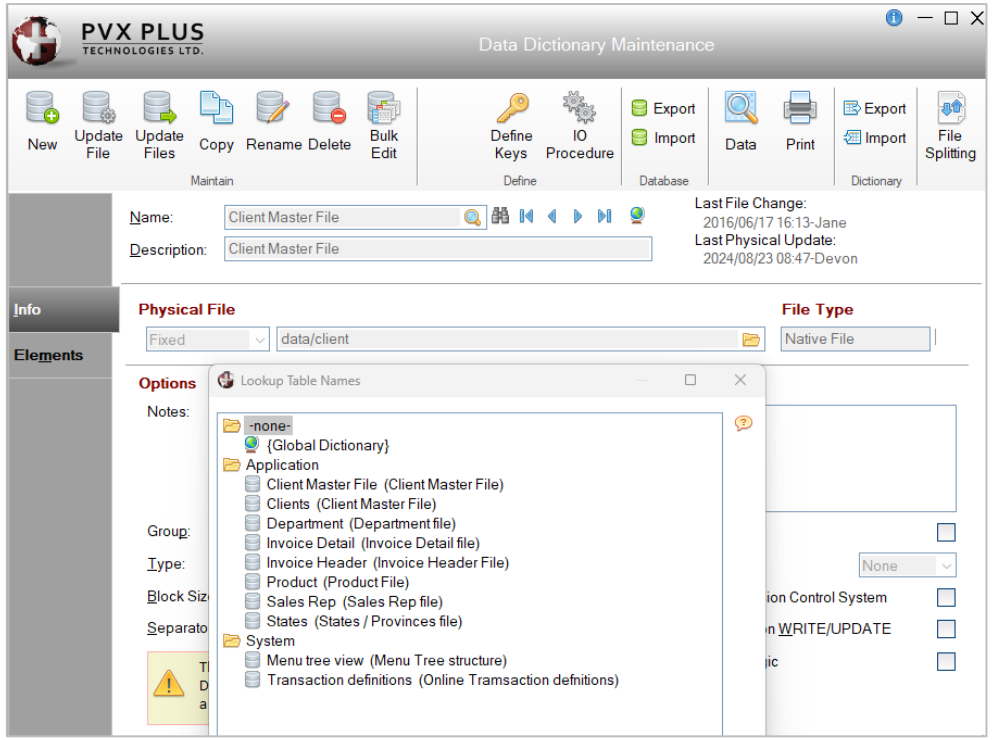
```
Call "*dict/sqlmake;LinkFile", fileName$, db_prefix$, connect$, db_name$, db_table$, db_user$, db_pass$, db_options$, link_file$, overwrite, stripPswd
```

HOW TO CONVERT TO EXTERNAL DATABASES

HOW TO CONVERT TO EXTERNAL DATABASES

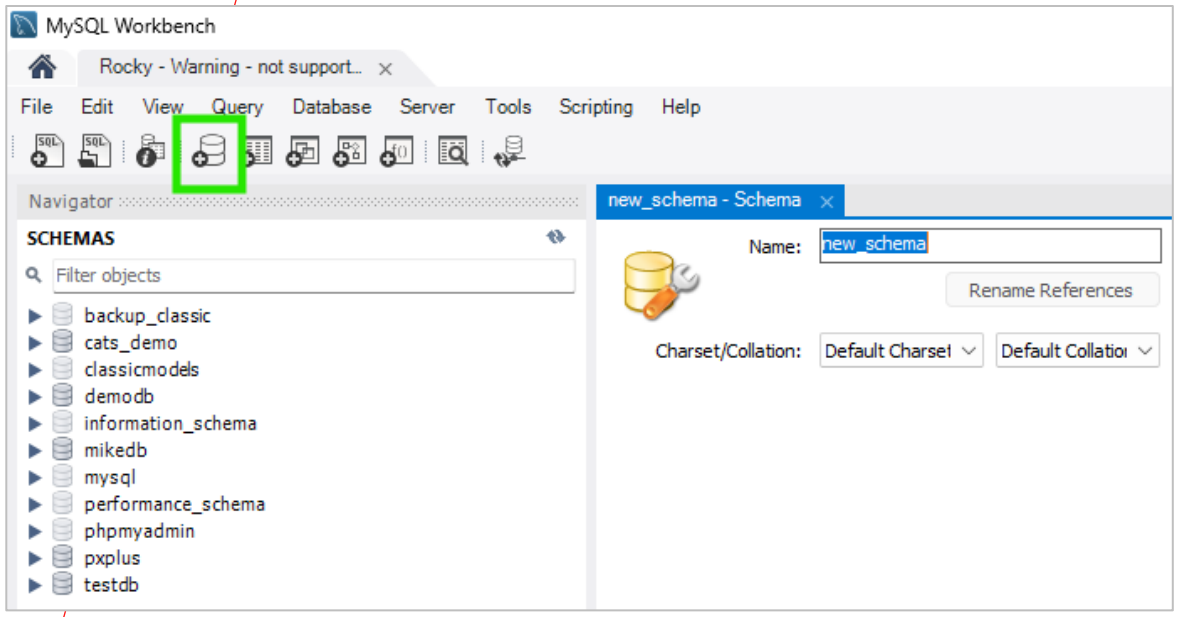
- Let's convert the Cats demo from using PxPlus native keyed files to using a MariaDB external database
- **Important:** Make sure you **back up** your PxPlus native keyed files before you start





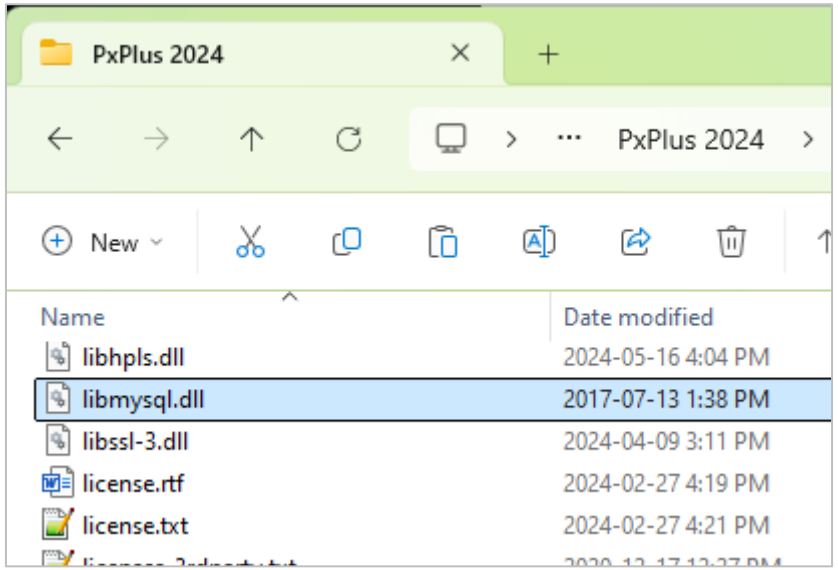
HOW TO CONVERT TO EXTERNAL DATABASES

- **Install and setup a database server if you don't already have one**
  - Use the software/tools provided by the database vendor to install and set up the database server
  - We used **MariaDB** (free MySQL)
- **Create a new database/schema on the server where the exported tables will go**
  - Use database management software or command line tools provided by database server to create the database/schema
  - We used **MySQL Workbench**

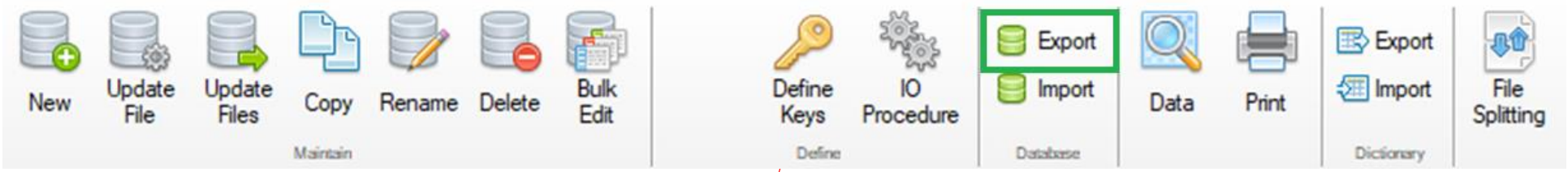


HOW TO CONVERT TO EXTERNAL DATABASES

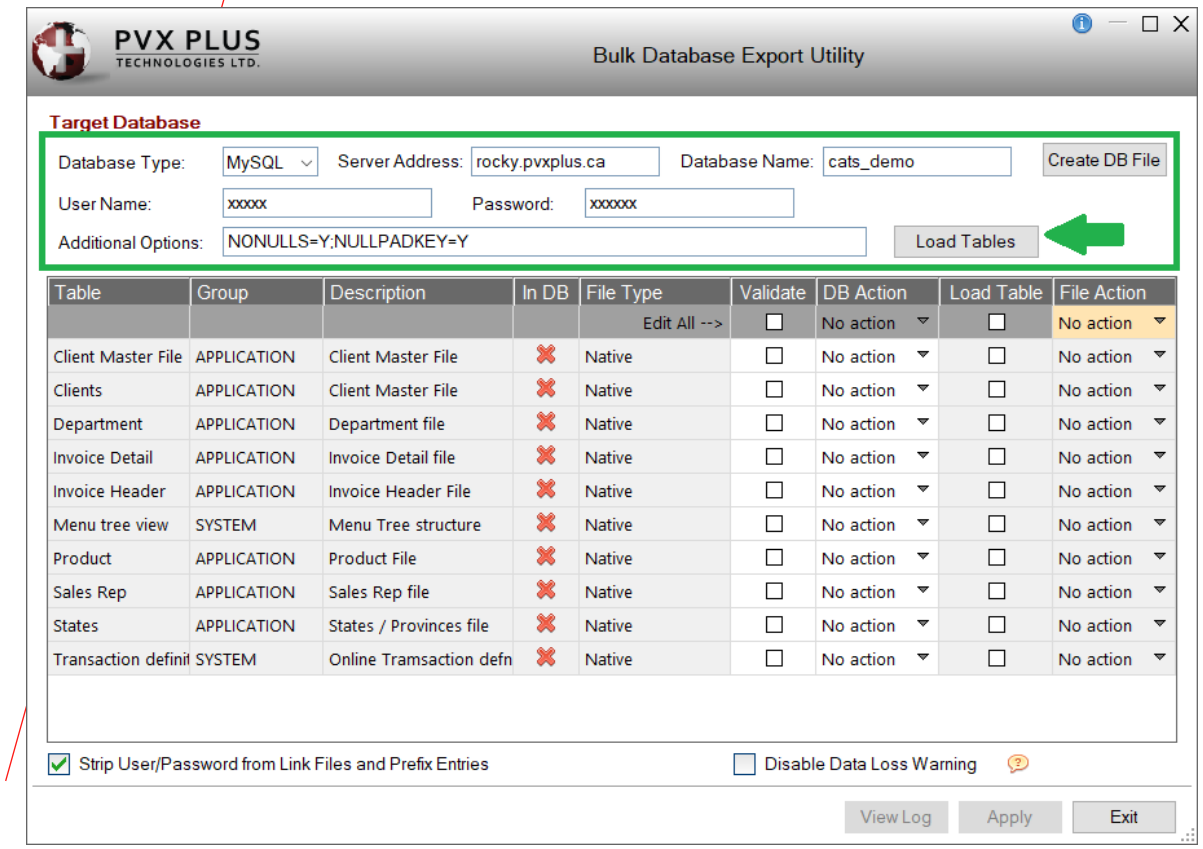
- **May need to install database connector**
  - [MYSQL], [DB2], and [OCI] interfaces all require a database connector
  - [ODB] requires a ODBC driver for the database and a data source name
- **Working with MariaDB, we need [MYSQL] interface so must download MySQL C Connector**
- **Install/Extract library file (libmysql.dll/libmysqlclient.so) to PxPlus install directory**
  - Use System Analysis Tool to test if [MYSQL] interface is working



HOW TO CONVERT TO EXTERNAL DATABASES

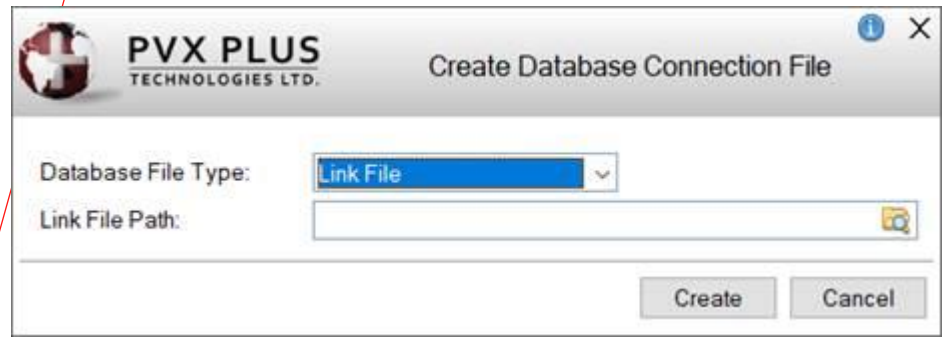


- **Open the Data Dictionary**
  - On the IDE, expand the Data Management category and click on Data Dictionary Maintenance
- **Launch the Bulk Database Export Utility**
  - In Data Dictionary Maintenance, click the Export button in the Database section of the top toolbar
- **Input the connection information to connect to your database (in our case, MariaDB)**
- **Click the Load Tables button**



HOW TO CONVERT TO EXTERNAL DATABASES

- **To prevent plain text login credentials and make it easier to change login credentials in the future:**
  - Click the **Create DB File** button to create a database Link File
  - Modify DB Link File to use a global variable for the User Name and Password
  - Add definition of User and Password and an OPEN of the DB Link File to the START_UP program
  - Password protect the START_UP program



Link File

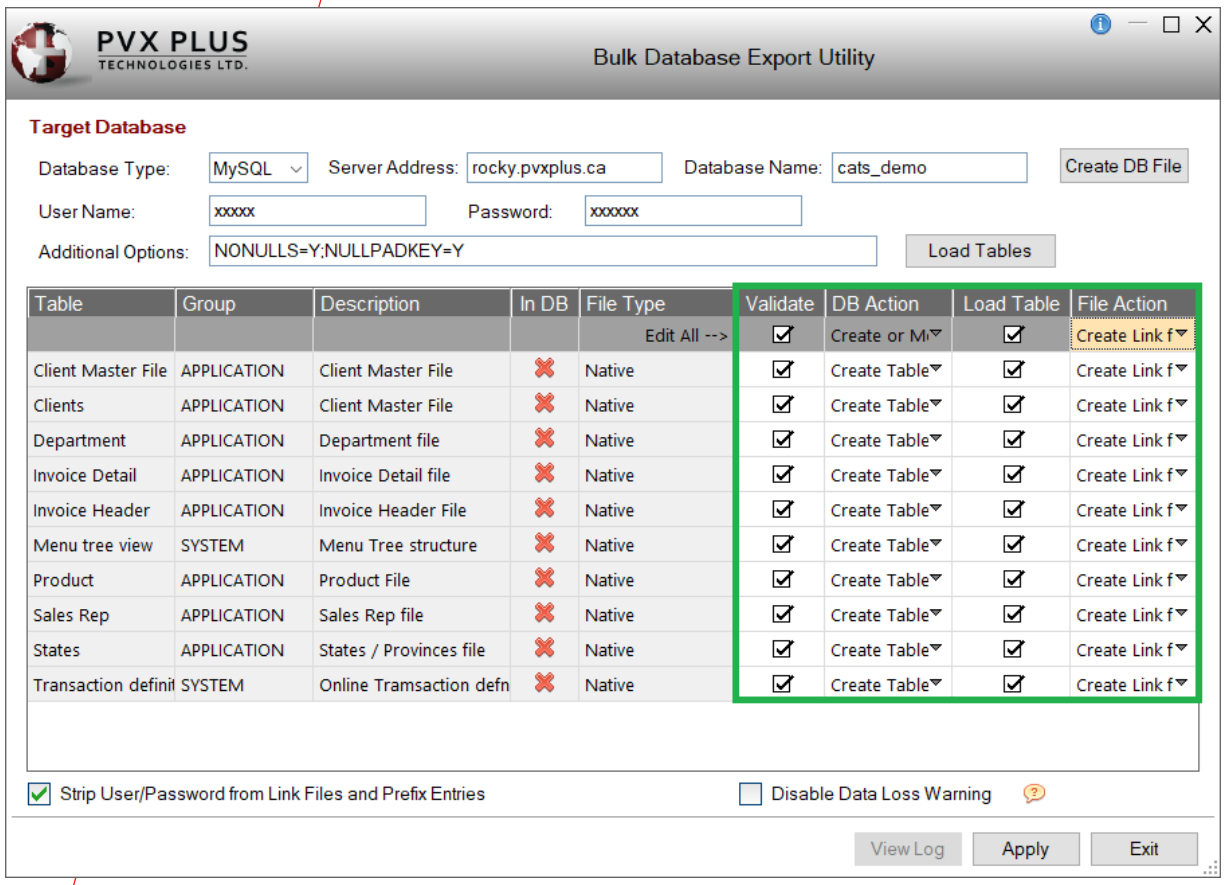
="USER="+%myDBUser$
="PSWD="+%myDBPswd$

START_UP

%myDBUser$="xxxxxx"
%myDBPswd$="xxxxxx"
open (gfn)myDBLinkFile
%myDB=lfo
%myDBUser$=""
%myDBPswd$=""

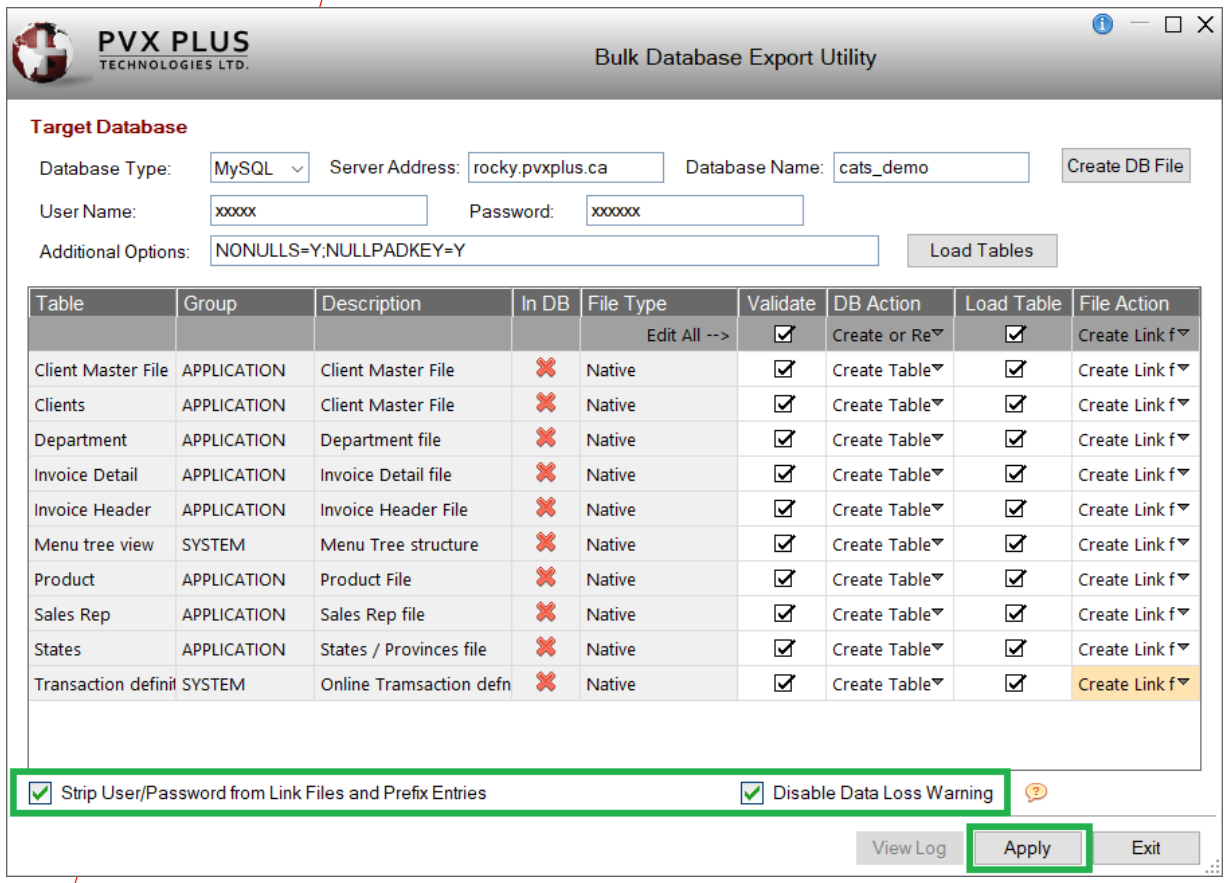
HOW TO CONVERT TO EXTERNAL DATABASES

- **Scroll through the list of tables from the data dictionary, and for all the tables you want to export:**
  - Click the **Validate** check box to ensure that the data in the file matches the data definition
    - If not a match, fix the definition or the data so that they do match; otherwise, the database will display an error when trying to load the data
  - Select the **Create Table** option from the **DB Action** drop-down list to create the table in the database
  - Click the **Load Table** check box to copy the data from the PxPlus native file to the database table
  - Select the **Create Link File** option from the **File Action** drop-down list to create a Link File to the table in the database, overwriting the PxPlus native file



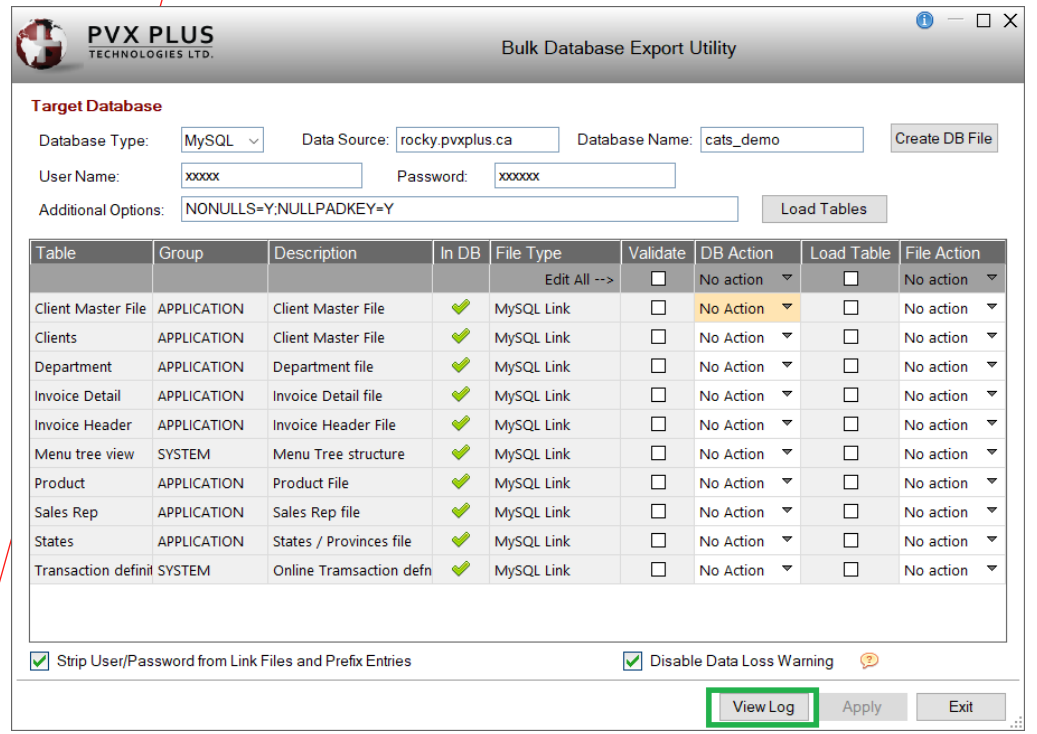
HOW TO CONVERT TO EXTERNAL DATABASES

- **Make sure that the Strip User/Password from Link Files and Prefix Entries check box is selected**
- **Select the Disable Data Loss Warning check box to avoid a warning for each table exported**
  - This is why we said earlier to backup your files so that you can always go back
- **Click the Apply button to perform the export**
  - When completed, a message box displays to list all the changes that were done



HOW TO CONVERT TO EXTERNAL DATABASES

- **Click the View Log button to view/save/print the log**
  - If any errors occurred, more details can be found in the log
- **Exit the Bulk Database Export Utility**




```
Export database Log - by Devon run 2024/08/23 12:21          Page:  1

1. Validate file (data/client) for: Client Master File

2. Replace database table record for: Client_Master_File in: [MySQL]cats_demo

3. Load data from file (data/client) for: Client_Master_File in: [MySQL]cats_
demo

4. Create link file (data/client) for: Client_Master_File in: [MySQL]cats_dem
o

5. Validate file (data/client) for: Clients

6. Create database table record for: Clients in: [MySQL]cats_demo

7. Load data from file (data/client) for: Clients in: [MySQL]cats_demo

8. Create link file (data/client) for: Clients in: [MySQL]cats_demo

9. Validate file (data/department) for: Department

10. Replace database table record for: Department in: [MySQL]cats_demo

11. Load data from file (data/department) for: Department in: [MySQL]cats_demo

12. Create link file (data/department) for: Department in: [MySQL]cats_demo
```

HOW TO CONVERT TO EXTERNAL DATABASES

- **Now if you look in the data dictionary, the files exported will display MySQL Link File as File Type**
- **Now if we open our Cats demo, it will work as before but the data is in the MariaDB database**