Altium Designer provides the ability to place components directly from a company database, by creating and using a Database Library. Placement is carried out from the Libraries panel which, after installing the database library, acts as a browser into your database.

After placement, design parameter information can be synchronized between placed components and their corresponding linked records in the database, using the Update Parameters From Database command. Full component updates - including the graphical symbol, model references, and parameters, can be performed using the Update From Libraries command. Parametric information from the database can also be included in the final Bill of Materials (BOM), ready for component procurement.

**Direct Placement - Beyond Mere Linking**

Altium Designer provides three methods of linking from a component to a database - using either a Database Link file (*.DbLink), a Database Library file (*.DbLib), or an SVN Database Library file (*.SVNDbLib) respectively. The DbLink approach provides an efficient means of linking and keeping the components used in your design (or libraries) synchronized with the data entered in the database. The DbLib/SVNDbLib approach takes this proven linking efficiency and adds to it the freedom to be able to place a component directly from the database - in essence creating the component dynamically from the information stored for it in the corresponding database record.

Before launching into the greater depths of the database library feature, it is worth discussing the differences between these methods, in regard to the way you use them.

**Linking using a Database Link file**

- Using this method, the Database Link file defines linkage between the schematic component and a matched record in a database. The record match is established by key field linking, which can be a single key field (for example a part number), or multiple key fields (by defining a Where clause).
- With this method of linking, the model and parameter information for the component must be predefined as part of the Altium Designer library component. The library component must also include the necessary key field information as part of it's definition. Once this has been defined you add a Database Link file to your Library Package, or Design project, then you can synchronize the component information (parameters) with the contents of fields in the database.
- Although each physical component defined by each database record does not need to map to a
unique Altium Designer library component - many database components can share the same component symbol - this method of linking would typically be used in a "one database record-to-one Altium Designer component" fashion. The unique Altium Designer component can either be an instance placed on a schematic sheet, or a unique component in a component library.

- With DbLink-style database linking, you include the Database Link file with the library package, or design project.

**Linking using a Database Library file**

- Using this method, the Database Library file also defines the linkage between the schematic component and a matched record in a database. Again, the record match is established by key field linking, which can be a single key field (for example a part number), or multiple key fields (by defining a Where clause).
- With this method of linking the component symbol, model, and parameter information for a component is stored as part of the record definition for that component in the external database. The referenced schematic component (in an underlying component library (*.SchLib)) is simply an empty shell, with a defined symbol only. There are no linked models, and no defined design parameters.
- When the component is placed, its parameter and model information is created on-the-fly, using the corresponding fields in the matched database record, and in accordance with defined mapping. One or more of these parameters will then be used to maintain an ongoing link back to the database, as per the matching criteria defined, enabling future synchronization after placement.
- This method of linking, due to its dynamic creation of components at the time of placement, lends itself very well to being used in a "many database records-to-one Altium Designer component" fashion.
- Unlike the DbLink-style of database linking, whereby the DbLink file must be included with the library package (or design project), the DbLib/SVNDLib file need not be added to the project. The resulting database library simply needs to be made part of the Available Libraries - accessible by the Libraries panel. Remember, the Available Libraries can consist of Project Libraries, Installed Libraries, or libraries found along specified search paths.
- You would typically configure DbLib/SVNDLib files in a library-oriented fashion. For example you might have one for all the resistors detailed in your company database, another for capacitors, and so on.

**DbLib vs SVNDLib**

An SVN Database Library is an extension of the Database Library model, with the difference being that the source symbol and models are stored under version control.

The source libraries are created, added-to, and maintained within, a Subversion-based repository. The link to the repository, and to the external database, is defined within an SVN Database Library file (*.SVNDLib). And since this approach is an extension of the Database Library, it is often referred to as a Version-Controlled Database Library.

**Librarian or Designer?**

There are essentially two modes in which you can work with a version-controlled database library - as a Librarian, or as a Designer.

As a Librarian, not only will you need to set up and maintain the external source control repository of symbol and model libraries, but you will also be responsible for setting up the SVN Database Library
file. This entails:

- Connecting to the external component database.
- Defining database record to component parameter/model mapping.
- Specifying the link to the SVN repository in which the symbol and model libraries are stored.

As a Designer, you will take the SVNDbLib file generated by the Librarian, and make it available in the Libraries panel - by adding it to the Available Libraries list - so that you can design with the bounty of database components within.

Altium Designer in no way prohibits a Designer from performing Librarian-based activities. Access rights/permissions defined for the source control repository will ultimately define how far the Designer can drive the Librarian 'car'. For example, the Designer may be able to checkout, open and modify a library, but is prevented from committing the change to the repository.

The use of these roles will differ from company to company. A larger company may have a dedicated library center which will, in the role of Librarian, solely be involved with the setup and maintenance of the symbol and model libraries in the source control repository, and the generation of the SVN Database Library file(s). The SVNDbLib file(s) will then be made available to the Designer(s) within that company, who will install and use the library files from which to place the components in their designs.

In a smaller company, both roles may be filled by the same person. Not only will that person setup and maintain the repository of libraries, but will also use the defined SVNDbLib(s) to carry out the design.

**The Source Control Repository**

The symbol and model library files need to be added to a Subversion (SVN) repository. Addition of such files and management of folder structure within the repository is performed using an SVN client, such as TortoiseSVN.

In Altium Designer, ensure that the SVN - Subversion provider option, on the Data Management - Version Control page of the Preferences dialog, is enabled.

One important thing to note when adding files to the repository, is that each symbol and model must be stored in its own library file. In a regular library - which can hold any number of symbols/models - changing a single entry would result in all being marked by the version control system as having been modified. Having one symbol/model per library file plays to the nature of version control, while allowing you to keep track of exactly what has been modified and what has not.

For the purposes of this document, it is assumed that the repository has already been created.

For more information on the use of the particular SVN client you are using, refer to the relevant documentation associated with that software. For more information on version control, see Version Control and Design Repositories. When reading this document, remember that for a version-controlled database library, the source control system used must be Subversion.
Using the Library Splitter Wizard

Dialog page: Library Splitter Wizard

If the libraries for the symbols and models are being newly created, there is no problem ensuring only one symbol/model per file. Typically however, the source libraries will already exist. To simplify the job of separating these libraries into single-entity files for addition to the SVN repository, Altium Designer provides a splitting tool - the Library Splitter Wizard. The wizard is accessible from the Tools menu when editing a PCB or Schematic library document.
Split source libraries into one symbol/model per file, using the Library Splitter Wizard.

Setting up for the splitting process involves the following steps:

1. Adding the required source library files.
2. Specifying the output directory. By default, the output directory will be set to the same directory in which the source library is stored. Care should be taken if the original source libraries are all stored in the same source directory folder. If the output directories are not
specified and are left at their defaults - pointing to the same source directory folder - you can end up with a very large mixture of single symbol and model libraries. Sifting through these when creating the repository structure may prove very time consuming.

As the split library files will be added to the SVN repository, it can be a good idea to setup the folder structure that you want in the repository - in terms of the symbol/model files - locally on your hard disk. Once the structure is defined, copy the source libraries to their respective folders within that structure. You can then just leave the output directory settings for each source library at their defaults. After the split, the generated single library files will already be in the correct locations. You can then simply delete the original source library from each folder and add the folders to the repository, using the SVN client you have decided to use.

3. Configure additional options to control how the source libraries are split. These include, for a schematic component library, options that allow you to strip the parameter and/or model information from each constituent component - leaving just the bare symbol.

4. Once the source libraries and options have been specified the wizard will perform an analysis, the result of which is the set of single symbol/model libraries that will be created, listed for you to review. The name of each split library to be generated is shown. The name used for a library will be the same as that used for the constituent component/model.

If any of the following characters have been used in the component/model name, they will be replaced by an underscore when creating the library name: "/", ";", ",", ":[", "]".

5. Review the list of libraries to be created and then click **Next** to proceed with the splitting process and ultimately obtain the libraries to be added to the source control repository.

**Creating the Database Library File**

The backbone of the database library feature is one of two files, depending on whether you are using a standard database library, or a version-controlled database library:

- **Standard Database Library** - the Database Library file. This file is created and managed using Altium Designer's DatabaseLib Editor. This editor becomes available when the active document in the main design window is a *.DbLib file.

- **Version-Controlled Database Library** - the SVN Database Library file. This file is created and managed using Altium Designer's SVNDatabaseLib Editor. This editor becomes available when the active document in the main design window is a *.SVNDLb file.
Example DbLib file, open in the DatabaseLib Editor. Hover over the image to see an example SVNDbLib file, open in the SVNDatabaseLib Editor.

Create a new file of these types by choosing the **File » New » Library » Database Library** command, or **File » New » Library » SVN Database Library** command, from the main menus.

For a standard Database Library, if a default document has been specified on the **System - New Document Defaults page** of the **Preferences dialog**, the Database Library file will be created using that default as a template. A different default can be assigned, for a Database Library file created as part of a PCB project, Integrated Library Package, or created as a free document.

Standard Database Library files, when added to a project, appear in the **Projects panel** under the **Settings\Database Library Files** sub-folder. Version-controlled Database Library files, when added to a project, appear in the **Projects panel** under the **Settings\SVN Database Library Files** sub-folder.

**Connecting to an External Database**

Table and mapping data will only appear in an Editor's main display window after the active Database Library/SVN Database Library file is successfully connected to the required external database. Connection is defined using the controls provided in the **Source of Connection** region.
Specifying the connection to the external database through a DbLib file. Hover over the image to see connection through an SVNDbLib file.

Any database which provides OLE DB support can be connected to. The options provided in this region each use an OLE DB connection string to connect to the target database. Some databases may not offer OLE DB support. However, virtually all Database Management Systems in use today can be accessed through the Open Database Connectivity (ODBC) interface. The database link feature uses Microsoft's ODBC provider, which allows an ADO (ActiveX Data Object) to connect to any ODBC data source. The result is that any ODBC database can be connected to. The OLE DB provider for the ODBC database is specified as part of the connection string.

Connection can also be defined on the Connection tab of the Database Connection dialog, accessed from the main Tools menu, or by clicking the Advanced button (to the right of the Connect button, at the bottom of the region).

**Fast Connection to Access and Excel Databases**

The Select Database Type option simply offers an expedited method of creating a connection string when the target database has been created using Microsoft Access, or Microsoft Excel. Using this option, simply select the database type and then browse to and select the required database file. The corresponding connection string will automatically be composed and entered into the field for the Use Connection String option.

The full path can be specified, or you can opt to store the path relative to the Database Library/SVN Database Library file.

**Building a Connection String**

If your company database is not Access or Excel-based, and you want to build the connection string explicitly, simply enable the Use Connection String option and then click the associated Build button to the right. The Data Link Properties dialog will appear.
The OLE DB Provider Microsoft Jet 4.0 OLE DB Provider is set by default on the Provider tab of the dialog and hence the dialog opens at the Connection tab. This is the default provider setting for new Database Library/SVN Database Library files and is also used to connect to Access database files (*.mdb). Change the provider as necessary.

From the Connection tab simply enter the name (including path) of the database you wish to connect to. Alternatively, use the button to open a dialog from where you can browse to, and open, the required file.

If your database requires login permission, enter this as required, along with any other advanced settings available from the Advanced tab of the dialog. The All tab provides a summary listing of link options defined, as well as extended options relating to the chosen OLE DB Provider. Options can be modified as required from this tab.

Once link options have been defined you can check for successful connection by clicking on the Test Connection button (on the Connection tab). A successful connection will yield a confirmation dialog to that effect.

The Data Link Properties dialog is a Microsoft dialog and, as such, pressing F1 or using the available Help button will gain access to the Microsoft Data Link Help file. This file is not part of Altium Designer's documentation set.

**Specifying a Data Link file**

If the data source to which you wish to connect is described using a Microsoft Data Link file (*.udl), simply enable the third of the connection options - **Use Data Link File** - and click the associated
**Browse** button to locate the required file. A Data Link File is essentially a storage vessel for a connection string.

**Proceeding with Connection**

After defining the connection to the external database, the text of the **Connect** button will become bold, signifying that you can proceed with the connection. If the connection details are correct, the table and mapping information for the target database will be loaded into the Database Library/SVN Database Library file. The text on the **Connect** button will change to Connected and the button will be grayed-out.

If there is a problem with the connection details, for example a connection string is built incorrectly, or a path is entered erroneously, connection will fail and a message will appear alerting you to this fact. Simply check your connection settings and click the **Connect** button again.

If you change the connection settings whilst connected to a database, live connection will be lost and the text on the **Connect** button will change to Reconnect. Click to re-establish the connection.

After successful initial connection, and after saving the Database Library/SVN Database Library file, the connection will be made automatically each time the file is opened, provided the target database's location and filename are not changed.

**Specifying the Link to the Version Control Repository (SVNDbLib Only)**

With a DbLib, the symbol and model libraries are stored on your hard disk, or other local/network medium. Remember that for an SVNDbLib, these libraries are stored under version control in a Subversion repository. As part of the SVNDbLib file, you must define the link to this repository. The connection to the SVN repository is defined on the **SVN Repository** tab of the Database Library Options dialog (Tools » Options). This dialog can also be accessed by clicking the **Options** button in the Field Settings region of the document view.
Database Table Listing

After successful connection to the external database, table and mapping data will be loaded. The left-hand side of the DbLib/SVNDbLib document lists all tables that exist in the connected database.

For an Excel-based database, a table is actually a sheet in that linked spreadsheet. The name of the table will have the suffix $.

If the target database has been created using multiple Excel spreadsheet files (*.xls), there is a limit of 64 sheets that can connect, due to ODBC driver limitations.

The **Enable** option next to each table entry allows you to control whether or not that table is to be part of the resulting database library. When the database library is added to the list of Available Libraries for browsing in the Libraries panel, each table will appear as a separate library entity. So although only one database library is added in real terms, from the perspective of the Libraries panel it is as though you have added multiple, distinct libraries. For more information, see Adding the Database Library.

As you click on a table entry in the list, its icon changes from to in order to distinguish it as being the currently active table. The table - with all its data - appears on the **Table Browser** tab of the document. This is an editable view of the table, and allows you to quickly refer to its contents, and edit where required, without having to launch the external database itself.
Browsing a source table in the connected database.

To enlarge the area of the Table Browser (as shown in the illustration above), simply collapse the entire area above (connection and field settings) by clicking the control, at the top-right of the document view. Click to expand. The table also supports familiar grouping, sorting, and filtering features.

Specifying Matching Criteria

After a component has been placed from the external database, there needs to be some way of retaining the link between that placed component and the database record whose information was used to create it. In essence, the two need to be matched.

When a component is placed, its parameter information is created on-the-fly, using the corresponding fields in the database record. The post-placement link between the schematic component and the database record is established using one or more of these parameters. The Field Settings region of the document allows you to define the matching criteria - either a simple, single key lookup, or a more advanced match using a Where clause.

Matching criteria is specified on a per-table basis.
If the **Single key lookup** option is enabled (default) the **Database field** and **Part parameter** fields become available. The former lists all of the available field names (column headers) in the active table of the database. As the parameters for a schematic component are added as it is placed, the **Part parameter** field will simply reflect whatever database field is chosen.

Typically, the lookup key field used is something that uniquely identifies each component in the external database, such as a Part Number. The chosen lookup field is distinguished on the **Field Mappings** tab of the document by the **Design Parameter** entry shown as grayed-out.

When using the **Update Parameters From Database** feature at some stage after placement, information is read from the chosen key parameter in the placed schematic components and then searched for in the chosen (key) field of the database - across all enabled tables. When there is a match, information from other cells in that record of the parent table can then be taken back to the
mapped parameters in the schematic component.

**Advanced Matching - the Where Clause**

While the *Single key lookup* option works well if there is a unique part number/id to match on, it is not so effective when matching by a parameter that is not unique, such as capacitance or resistance. In this case the more advanced Where clause should be used, enabling you to specify multiple key matching in order to link the schematic component to its source database record.

In its simplest form the Where clause (written using SQL syntax) reflects the chosen entries that define the single key lookup. For example if the **Database field** was chosen to be **Part Number** - the default - the **Part parameter** field would automatically be set to **Part Number** also, and the entry for the Where clause would be:

\[
\text{[Part Number]} = '{Part Number}'
\]

The square brackets around the database field (table column) are quote characters, as specified on the **Advanced** tab of the **Database Connection dialog**. Access this dialog either by clicking on the **Advanced** button in the **Source of Connection** region of the document, or from the main **Tools** menu.

The square brackets around the database field (table column) are quote characters, as specified on the **Advanced** tab of the **Database Connection dialog**. Access this dialog either by clicking on the **Advanced** button in the **Source of Connection** region of the document, or from the main **Tools** menu.

When quoting tables, the specific quote characters used will depend on the database you are using. For example, square brackets [ ] are only usable in Microsoft databases like Access, Excel via ADO, or MSSQL (later versions). MYSQL would use the ' character for quoting. You really only need to quote column names, in any database, if they include spaces or are reserved words (for that database). Check the documentation for your particular database software to see which quote characters are used (if any).

The curly brackets (braces) specify that the entry being referenced is a design parameter. The single quotes are used to specify the design parameter be treated as a string, as opposed to a number (no quotes). The type matching is very important, as SQL is type sensitive. The design parameter should be made the same type as the column in the database.

Using standard SQL syntax, the Where clause can then be extended to match using multiple **Database field/Part parameter** entries, for example:
In this case a single record in the relevant table of the database would be linked to, using three different design parameters. Notice that the entry for the Tolerance design parameter is not quoted. This means that the column type in the associated table of the database is Number and not String.

Using standard SQL syntax you can conceivably make the Where clause as simple or as complicated as you like.

**Mapping Database Fields to Design Parameters**

Design parameters for a component placed from a database library are created/added at the time of placement. Which parameters are actually created, and the options used to update their information after placement - using the Update Parameters From Database command - is determined by mapping and update information specified in the Database Library/SVN Database Library file. These settings are performed on the Field Mappings tab of the main document view.

Mapping and update options are specified on a per-table basis.

![Field Mappings Table Browser](image)

Specify parameter mapping and update options on the Field Mappings tab of the DbLib/SVNDblib document.

**Model and Parameter Mapping**

The first two columns (from the left) on the Field Mappings tab are used to control which information from the database is to be mapped to the component's attributes, models and parameters.

- **Database Field Name** - this column lists all field (column) names in the currently active table
of the database.

- **Design Parameter** - this column defines how each corresponding field in the database is to be used. This could be to source a component's symbol or footprint model (it will be enclosed in square brackets), or if it is to be included as a component parameter (these are not enclosed in brackets).

**When you open a DbLib/SVNDbLib document that has been connected to a database, you will notice that some Design Parameters include square brackets and others do not. The square brackets denote a reserved name, such as `[Library Ref]`. This data is used to populate the component's attributes and models. Design Parameters that do not include square brackets become component parameters.**

Initial mapping is performed automatically upon connection to the database, with all database fields mapped.

Note that the automatic mapping assumes that the Database Field Names match the reserved name used in Altium Designer. If they do not, then the mapping must be manually configured.

**Attributes & Models**

If the database field name is one of the following reserved names, the corresponding attribute/model mapping entry will be automatically set in the **Design Parameter** field:

- **Description** → `[Description]`
- **Footprint Ref** → `[Footprint Ref]`
- **Footprint Path** → `[Footprint Path]`
- **Footprint Ref n** → `[Footprint Ref n]`
- **Footprint Path n** → `[Footprint Path n]`

Unlimited footprint model references (and paths) can be specified in a database table and mapped in the DbLib/SVNDbLib file. In the reserved names, `n` represents a positive integer starting from 2.

- **Library Ref** → `[Library Ref]`
- **Library Path** → `[Library Path]`
- **Orcad Footprint** → `[Orcad Footprint]`
- **Orcad Library** → `[Orcad Library]`

These references are used when importing an OrCAD CIS configuration file and libraries, and generating a standard database library (DbLib).

- **PCB3D Ref** → `[PCB3D Ref]`
- **PCB3D Path** → `[PCB3D Path]`

References to PCB3D refer to the legacy 3D viewer, these should not be used for new designs. The PCB Editor's native 3D engine can render component bodies and imported 3D model files associated with component footprints.
PCB3D model mappings can be defined in an SVNDbLib file, however storage of PCB3D model libraries in the Subversion repository is not supported.

- **Sim Description** → [Sim Description]
- **Sim Excluded Parts** → [Sim Excluded Parts]
- **Sim File** → [Sim File]
- **Sim Kind** → [Sim Kind]
- **Sim Model Name** → [Sim Model Name]
- **Sim Netlist** → [Sim Netlist]
- **Sim Parameters** → [Sim Parameters]
- **Sim Port Map** → [Sim Port Map]
- **Sim Spice Prefix** → [Sim Spice Prefix]
- **Sim SubKind** → [Sim SubKind]

Only one simulation model link can be defined for a component in an external database. Typically there will only ever be a single simulation model linked to a component. Should you wish to set up multiple simulation model links, the other links will need to be defined and stored with that component in the source schematic library file.

Linked simulation models are supported for a version-controlled database library (SVNDbLib), however storage of simulation model files in the Subversion repository is not supported.

These mappings define the component attribute and model information for the component. When the component is placed, the schematic symbol specified by the corresponding database record's [Library Ref] field will be extracted from the specified schematic library. Similarly, PCB footprint and Simulation model information stored in the record will be added to the component as linked footprint and simulation models respectively.

The [Library Ref] entry must exist in the **Design Parameter** column and be mapped to the **Database Field Name** that specifies the schematic symbol, to be able to place a component from the Database Library/SVN Database Library onto a schematic. If the database table contains the symbol reference under a different **Database Field Name**, for example SCH Symbol, you will need to manually set the associated **Design Parameter** entry for this field to [Library Ref], using the available drop-down list for that cell.

Similarly, if model reference information is entered into the database using different field naming, you will need to manually map by choosing the appropriate **Design Parameter** entry ([Footprint Ref], [Footprint Ref n], [PCB3D Ref], [Sim Model Name], and so on) from the drop-down list, for each field in turn.

In order to define the symbol and model information for a
component upon placement, it is the symbol and model reference fields in the database that are the crucial mapping entities - ensure that there is a symbol and at least one PCB footprint reference as part of the defined mapping.

As mentioned, multiple PCB models can be mapped. The **Database Field Name** that is mapped to **Design Parameter** [Footprint Ref] will be the default footprint when the component is placed on the schematic. It is this footprint that will be placed when the design is transferred to the PCB domain.

When mapping symbols and models for a DbLib/SVNDbLib file, path information in the external database is optional. For more information see [Specifying Symbol and Model Library Locations](#).

**Parameters**

All other database field names will be automatically mapped to design parameters using the same names. For example, if a field in the database is called Tolerance, a design parameter with the name Tolerance will be mapped to it. You can change the name for a design parameter simply by clicking in its cell and typing the new name directly. It is these design parameter names that will appear in the **Parameters** region of the component's associated properties dialog, once it has been placed on a schematic sheet.

You may have a large number of data fields associated with a component in the database, not all of which you will want, or even need, added as design parameters to the component when placed on a schematic sheet. Much of this information may only be required when generating a Bill of Materials. The **Report Manager dialog** includes an option that allows you to add parameter information to a BOM, directly from a linked database - allowing you to reduce the amount of information that gets 'carried' with the schematic source documents. For more information, refer to the section [Adding Database Information Directly to a BOM](#).

For fields that you explicitly do not want mapped from the database, set the **Design Parameter** entry to [None]. Unmapped database fields are distinguished on the tab by the use of a red cross icon (❌). Mapped database fields are distinguished by a green tick icon (✔).

Configure parameter mapping as required. Set to [None] to prevent a parameter in the database being mapped to the placed component. This information can be included directly in the BOM.

To quickly remap an unmapped field, click inside the row for that field and use the **Ctrl+D** keyboard shortcut. Note that for model mappings you will need to manually select from the associated **Design Parameter** drop-down.
Parameter Update Options

The remaining columns on the **Field Mappings** tab allow you to specify the actions taken for parameters, when placing a component from the database library for the first time, or updating a component after it has been placed using the [Update Parameters From Database command](#).

The four columns are as follows:

- **Update Values** - a cell in this column is used to determine the action that should be taken if the parameter exists both on a schematic sheet and in the database, but the values are currently different. Choose to update the parameter of the placed component with the value stored in the database (Update), or not to update at all (Do not update). This option is obeyed when using the [Update Parameters From Database command](#), after the component is placed.

- **Add To Design** - a cell in this column is used to determine the action that should be taken if the parameter is found in the database, but does not exist for the placed component. You can choose to add the parameter (Add), not add the parameter (Do not add), or add the parameter only if it has a value assigned to it in the database (Add only if not blank in database). This option is obeyed both when initially placing the component from the database library and when using the [Update Parameters From Database command](#), after the component has been placed.

- **Visible On Add** - a checkbox in this column is used to determine whether a newly added parameter, resulting from initial placement or update after placement, is made visible for the component on the schematic sheet (enabled) or not (disabled).

- **Remove From Design** - a cell in this column is used to determine the action that should be taken if the parameter is found to exist for the placed component, but either is not in the database, or is, but has no value. You can choose to not remove the parameter at all (Do not remove), or only remove it if it has no value assigned to it in the database (Remove only if blank in database). This option is obeyed when using the Update Parameters From Database command, after the component has been placed.

Initially, the **Update Values**, **Add To Design**, and **Remove From Design** fields - for each mapped database field - will be set to the entry Default, and the **Visible On Add** option will be disabled, as illustrated in the following image.
Looking at the image, there are four important points to make regarding update options:

1. Unmapped database fields will have no associated update options.
2. Attribute and Model based mappings (entries with square brackets in the Design Parameter column) will have no associated update options, as these are not design parameters.
3. The key field (e.g. Part Number in the image) will have no associated update options. This field is solely used for parameter matching purposes.
4. A setting of Default causes an update option to follow its corresponding default definition, as specified on the Default Actions tab of the Database Library Options dialog (Tools » Options). This dialog can also be accessed by clicking the Options button in the Field Settings region of the document view.
Define default parameter update options in a central location for a DbLib. Hover over the image to see the equivalent for an SVNDbLib.

The fourth point is beneficial in that it allows you to specify update options from a central location, and then point to that location when defining the update options for each mapped field. That is why the Default entry is loaded automatically into the relevant update fields upon mapping a database field to a design parameter.

Should you wish to override the default setting for an update option, simply click inside the relevant update field, on the Field Mappings tab, then click again to access a drop-down providing the applicable update choices.
Parameter update options can be manually overridden, if required.

In this way, you have full control over how the parameters in the design are updated. You can set all fields to Default and make the required update decisions from the **Database Library Options dialog**, set each update field individually, or have a mixture of the two - the decision is entirely yours to make as you see fit. For placed components the update, when performed, is carried out through use of an **Engineering Change Order dialog**. If at this stage there are updates that you would prefer not to make, you can simply opt to not include those particular changes - giving you the final, and ultimate say, in which design parameters get updated.

### Specifying Symbol and Model Library Locations

When you place a component from a database library its symbol - specified by the [Library Ref] mapping - is extracted from the specified schematic library (*.SchLib). Similarly, any model references (footprint, PCB3D, simulation) specified in the database will reside in underlying PCB Library (*.PcbLib), PCB3D Library (*.PCB3DLib) and Simulation Model (*.mdl, *.ckt) files. The paths to these files can be specified explicitly in the database by:

- Entering an absolute path to the file.
- Entering a relative path to the file.

If you have defined fields in your database for path information, these fields need to be mapped to the appropriate design parameters - [Library Path], [Footprint Path], [PCB3D Path], [Sim File], and so on (refer back to **Model and Parameter Mapping**).

Entering paths - even relative - in a database table can be a little restrictive. If you move the location of a library or model file, you would need to update the database table accordingly. To give you greater freedom, you have the ability to specify library search information within the DbLib/SVNDbLib file itself. This allows you to specify the name of the source library or model file in the database, or not to define it at all.
**Using Search Paths with a large number of symbol/model files** - While search paths offer greater flexibility and simplicity for configuring how the symbols and models are located, this approach is not recommended when there are a large number of symbol or model files present in the search location. Depending on the file naming scheme used, Search Paths can require that every library/model file present in the path is searched every time a symbol/model is required, for example when browsing components or transferring the design from schematic to PCB.

For a Database Library

For a DbLib, library search paths are defined on the **Symbol and Model Search Paths** tab of the **Database Library Options dialog** (Tools » Options). This dialog can also be accessed by clicking the **Options** button in the **Field Settings** region of the document view.

![Database Library Options dialog](image)

Specifying search paths for symbol and model libraries.

To add a path to the list:

1. Either type the path directly into the available field, or click on the **Browse for Folder** button to access the **Browse for Folder** dialog, from where you can locate the directory in which the required library/model file(s) reside.
2. You can add either as a full path, or as a relative path (relative to the location of the DbLib file). Control this using the **Add/Update As Relative Path** option.
3. After specifying the required path, add it to the search paths list by clicking the **Add** button.
4. Continue specifying additional search paths as required.

For direct entry, if you specify an incorrect path (e.g. to a folder that does not exist) the
entry can still be added but will appear grayed-out in the list, to distinguish that it is an invalid search path. If you find that you have entered a path erroneously, you can select it in the list and either click the **Remove** button, or modify its path definition and click the **Update** button.

The library search paths determine where library and model files can be found when placing form the database library and when searching for a model after placement. The specific schematic symbol, footprint model(s), PCB3D model(s), or simulation model used, will depend on how you have set up your library search paths and whether you have added specific library information into your database. The searching will proceed in the following order:

1. If a full path exists in the mapped path field for the symbol or model, use that library/model file and extract the symbol or model specified in the applicable reference field.
2. If a relative path exists in the mapped path field for the symbol or model, use that library/model file and extract the symbol or model specified in the applicable reference field.
3. If only a library/model file name exists in the mapped path field for the symbol or model, use the search paths to locate the first library/model file that matches the specified name and which contains a match for the symbol or model specified in the applicable reference field.
4. If no library/model file information exists in the database, use the search paths to locate the first library/model file containing a match for the symbol or model specified in the applicable reference field.

**For an SVN Database Library**

For an SVNDbLib, the base directories within the Subversion repository - in which the symbols and footprint models reside - are specified on the **SVN Repository** tab of the **Database Library Options** dialog (Tools ➤ Options). This dialog can also be accessed by clicking the **Options** button in the **Field Settings** region of the document view.
Specifying base repository directories for symbols and footprints.

Use the available fields in the **Models Location** region to specify the base directory for the symbols and footprints. Click the **...** button at the right of a field to access the *Browse for directory* dialog - a window into the linked repository. Use this dialog to select the required folder.

It is important to stress that the symbols and footprints must reside within the base repository directories specified. They can of course be in sub-folders of those directories and the paths specified for both symbols and footprints can point to the same directory in the repository.

The model locations determine where library and model files can be found when placing from the version-controlled database library and when searching for a model after placement. The specific
schematic symbol and footprint model(s) used will depend on how you have set up these locations, and whether you have added specific library information into your database. The searching will proceed in the following order:

1. If a full path exists in the mapped path field for the symbol or footprint model (e.g. http://MyServer/svn/MyCompany/ParentDirectory/SchematicSymbols/Capacitor_NonPolarized.SchLib), use that library file and extract the symbol or footprint specified in the applicable reference field.

2. If a relative path (relative to the root of the repository) exists in the mapped path field for the symbol or footprint model (e.g. /ParentDirectory/SchematicSymbols/Capacitor_NonPolarized.SchLib), use that library file and extract the symbol or footprint specified in the applicable reference field. The URL for the repository - specified in the Repository Server Connection region of the dialog - will be prefixed to the path you enter, to give the absolute address.

3. If only a library file name exists in the mapped path field for the symbol or footprint model (e.g. Capacitor_NonPolarized.SchLib), use the specified model locations within the repository to locate the first library file that matches the specified name, and which contains a match for the symbol or footprint model specified in the applicable reference field.

When searching for a symbol/model match, the flattened folder paths in the base symbol or footprint directory are sorted and searched alphabetically. If the Library Splitter Wizard is used, there will always be a corresponding library with the name of the actual symbol/footprint.

4. If no library file information exists in the database, use the specified model locations to locate the first library file containing a match for the symbol or footprint model specified in the applicable reference field.

When locating the symbol/footprint, the system will initially look for the first library named like the symbol or footprint itself. For example, if the logical symbol name in the database (specified by the [Library Ref] mapping) is Capacitor_NonPolarized, the system will look for the first file named Capacitor_NonPolarized.SchLib and look for the symbol within this file. If the symbol/footprint cannot be found in this way, the system will look for a match in all libraries.

Modifying a Database Table

The occasion will often arise where a change is required to a component record in the database. An existing component may need one or more of its fields updated to better reflect its current status. A new component may be available for use and therefore needs to be added to the relevant database table. Supply chain information may need to be added to components. A component may have been marked as 'obsolete' and, no longer to be used in designs, is required to be deleted from the database altogether.

Both the DatabaseLib Editor and the SVNDatabaseLib Editor enable you to perform these actions directly, through the Table Browser, eliminating the need to open the external database. With the required table selected, commands for manipulating its entries in the database are available from the right-click menu.
Accessing commands to modify the selected table in the linked database.

**Editing an Existing Component**

An existing component record in a linked database table can be modified in two ways:

- **In-place Editing** - simply click within the required field and modify its value as required. Press **Enter**, or click away from the field when done, and click **Yes** in the subsequent dialog that asks you to confirm committal of the change (commiting it to the database).
- **Dialog-based Editing** - simply right-click on the entry for the required component record, and choose the **Edit Component** command from the context menu. An **editing dialog** for that component will appear, listing all of the fields for the parent database table, along with their values. Click on a **Field Value** entry to edit it. Click **OK** when you have finished modifying the desired fields, to commit the changes back to the database.

Demonstrating component record field editing, using the two editing methods.
Adding a New Component

To add a new component to a database table, right-click anywhere within the **Table Browser** and choose the **Add New Component** command from the context menu. The **New Component dialog** will appear, with all of the fields for the parent database table listed. Simply enter the information for this new physical component as required, then click **OK**. The new component will be added to the bottom of the table.

Demonstrating the addition of a new component to a table in the linked database.

Removing an Existing Component

To permanently remove an existing component from a database table, right-click on its entry in the **Table Browser** and choose the **Delete Component** command from the context menu. A confirmation dialog will appear - click **Yes** to proceed with the removal.

Managing Supply Chain Information

**Related page:** [Linking to Supplier Data](#)

Using Altium Designer's **Live Supplier Data feature**, you have the ability to search supplier databases from within Altium Designer, and link your design components to matching Supplier Items. For any given Supplier Item found through a search, you can also import its parameters, data sheet links, pricing and stock information, as parameters of a target component in a DbLib/SVNDbLib). In addition, you can also create whole new components in the targeted database table - directly from Supplier Items! The linked supplier data can be included at design release time, in a Bill of Materials - especially useful from a project costing and procurement perspective.

At the heart of this feature is the **Supplier Link**. This fundamental entity provides the live link between an Altium Designer component and an item in a Supplier's database of electronic components.

Creation of a Supplier Link involves the addition of two parameters to the target Altium Designer
component involved:

- Supplier n
- Supplier Part Number n

n is an integer suffix, used to distinguish between Supplier Items, where multiple links are added to the same component.

**Adding Supplier Links**

Creating a Supplier Link is a simple, straightforward process. In a manual fashion, this process is facilitated most easily by using the [Supplier Search panel](#) to search for the required Supplier Item. Once found, simply click on that item, in the results region of the panel, then drag-and-drop onto the required component record, on the [Table Browser](#) tab of the active DbLib/SVNDDbLib file.

![Supplier Search panel](image)

Demonstrating the addition of a Supplier Link to a component record in the linked database.

And by using the Supplier Search panel, a Supplier Link can also be created automatically in one of the following ways:

- By right-clicking on the Supplier Item entry in the results region of the Supplier Search panel, and choosing the [Add Supplier Link To](#) command. With the DbLib/SVNDbLib file active, the Supplier Link will be added to the focused component record.
- By right-clicking on the Supplier Item entry in the results region of the Supplier Search panel, and choosing the [Add Supplier Link And Parameters To](#) command. This command is used to add parameters, data sheet links, pricing and stock information for the selected Supplier Item.
Item, along with a Supplier Link to that item, all in the one step. With the DbLib/SVNDbLib file active, the parameters and Supplier Link will be added to the focused component record.

- When creating a new library component, by right-clicking on the Supplier Item entry in the results region of the Supplier Search panel, and choosing the **Import Into LibraryName As ComponentName / PartNumber** command. This command is used to create a new component in the target library DbLib/SVNDbLib directly from the selected Supplier Item. The ComponentName/PartNumber for the new component will be named using the Supplier Item’s Manufacturer Part Number parameter. A Supplier Link to the item will be added to the component record, along with all parameters for that item (parameters, data sheet links, pricing and stock information).

**Viewing/Editing Supplier Links**

The Supplier Links currently defined for a database component can be viewed and edited using the **Supplier Links dialog**. To access this dialog, right-click inside the record for the component, on the **Table Browser** tab of the active DbLib/SVNDbLib file, and choose the **Edit Supplier Links** command from the context menu.
An example of accessing the currently defined Supplier Links for a selected database component.

The dialog lists all of the currently defined links for the component. The upper pane shows information for added supplier items, including Supplier, Description, and Unit Price. You can add or remove supplier items to/from this area. The lower pane shows the detailed Parameters, Documents, Pricing and Stock information for a selected item in the upper pane.

To add an additional Supplier Link, simply click on the Add button. The Add Supplier Links dialog will appear, from where you can search for, and add a link to, the required item.
Supplier Links as required, using the Add Supplier Links dialog. Searching for Supplier Items is performed in the same way as when using the Supplier Search panel.

To delete a Supplier Link for the component, simply select its entry, in the upper region of the Supplier Links dialog, then click the Remove button.

Many suppliers offer discounts for bulk orders. Use the Order Quantity field to see how pricing changes, per unit, based on quantity of item ordered from the supplier. In addition, you can also change the currency used for the Unit Price - simply select from a range of supported currencies in the drop-down field to the left of the Order Quantity field. Currency conversion will use the latest daily rates, provided you are signed in to your Altium account.

Modifying Symbols and Footprints

Besides maintaining the content within the database itself, there is also a requirement to maintain the referenced symbols and models, as and when necessary. This involves making modifications to the schematic symbols and the footprint model(s). And for a standard database library (DbLib), the definitions in simulation model files.
Both types of database library - standard and version-controlled - provide the ability to access the referenced symbol and footprint model for editing, directly from within the main DbLib/SVNDbLib document view when open as the active document. The following sections take a look at how to do so, which is a little more involved, but also a little more powerful, when using an SVNDbLib.

Note that only the symbol and default footprint for a component record can be accessed for editing through the respective (and active) database library file. Remember that the default footprint model is the one entered into the **Database Field Name** that is mapped to **Design Parameter [Footprint Ref]**.

**When using a DbLib**

To open a component's referenced symbol/footprint model library file for editing, simply right-click on the component's entry in the **Table Browser**, then select the appropriate command - either **Open Symbol**, or **Open Footprint** - from the context menu. The applicable library file will be opened and the symbol/footprint presented as the active entity in the workspace, ready for you to perform your edits.

The library opened will be the one in which the referenced symbol/footprint is found - either specifically, if library information is present in the database, or if this is not present, by using the defined paths on the **Symbol and Model Search Paths** tab of the **Database Library Options dialog**. Refer back to the section on **Specifying Symbol and Model Library Locations for a Database Library**.

Once changes have been made as required, simply save the library file.
Example of accessing a database component's referenced symbol, directly from within the DbLib. Hover over the image to see access of the same component's referenced footprint model.

The source library file could also be accessed outside of the DbLib file. Either use the library editor's File » Open command to browse to and open, the required library, or drag-and-drop the library file from Windows Explorer directly into Altium Designer.

When using an SVNDbLib

As mentioned, the ability to access and edit the symbols and models referenced by database components, and accessed through an SVNDbLib file, is a little more complex, but also more powerful. Modifications are made to local working copies of the libraries stored in the SVN repository.

Specifying a Local Workspace

To make modifications to component symbols and footprint models, you will need to check the files out from the repository into a local working directory. Specify the required directory to be used on the
Data Management - SVN Libraries page of the Preferences dialog. A default working folder location is defined - \Users\<ProfileName>\Documents\My Designs\SVNDLib Working Folder. Change this as required by either clicking the button to browse to an existing folder, or by entering a path to the folder directly.

The folder nominated as the local working directory should only be used to contain a copy of the version-controlled files. Other files, such as *.SVNDbLib, *.mdb, and *.xls, should be located in a different directory, as these are not under version control.

The default working folder location does not physically exist by default. This will be created when checking out files from the repository. This will also be the case if you specify a folder (through direct path entry) that does not currently exist.

To quickly check out all symbol and footprint model libraries from the repository into the nominated local working folder, simply run the Check Out All Working Folders command from the SVNDatabaseLib Editor's main Tools menu. The folders checked out will be those specified in the Models Location region, on the SVN Repository tab of the Database Library Options dialog.
Making Changes

Once the symbol and footprint model libraries have been checked out to your designated local working folder, you can then make modifications as and when required. This involves making modifications to the local copy of the schematic symbols and footprint models, then committing these changes back to the SVN repository.

As with a DbLib, the following methods of accessing a symbol/model for editing are available:

- Within the SVNDbLib document, right-click on the component's entry in the Table Browser, then select the appropriate command - either Open Symbol, or Open Footprint - from the context menu.

The local library opened will be the one in which the referenced symbol/footprint is found - either specifically, if library information is present in the database, or if this is not present,
by using the defined model locations on the **SVN Repository** tab of the **Database Library Options** dialog. Refer back to the section on **Specifying Symbol and Model Library Locations for a Database Library**.

- Once the SVNDbLib is added to the Available Libraries list, you can access edit commands directly through the **Libraries panel**. Simply select the required component entry, then use the **Edit Symbol** command (on the right-click context menu for the **Components** region) or **Edit Footprint** command (on the right-click context menu for the **Models** region), respectively.
- Use the library editor's **File » Open** command to browse to and open, the required library (in your local working folder).
- Drag-and-drop the library file (in your local working folder) from Windows Explorer directly into Altium Designer.

In each case, once changes have been made as required, simply save the library file.

However, the main interface between the folders and files checked out to the local working folder, and the repository itself, is the **Storage Manager panel**. When the SVNDblib file is the active document, the **Storage Manager** panel will populate with the symbol and model folders checked out to the local working folder. The exact folders listed will be those specified in the **Model Locations** region, on the **SVN Repository** tab of the **Database Library Options** dialog.
The Storage Manager panel reflects the local working folder - provided the SVNDblib file is the active document within Altium Designer.

To open a library file for editing, simply right-click on its entry in the All Files region of the panel, and choose Open from the context menu that appears. Make changes as required and save the file as normal using the associated Editor's standard Save command.

Committing Changes to the Repository

Panel page: Storage Manager

Although there are different ways to open a library for editing, committal of any changes made - from within Altium Designer - can only be carried out through the Storage Manager panel.

Within Altium Designer, the controls for committing to, and updating from the repository, are solely available through the Storage Manager panel. Of course, you can also commit, update, and check for modifications outside of Altium Designer, using the features and functionality of your external SVN client.

One of the most important things to remember when using the Storage Manager panel, is to refresh
the status of the link between the local working folders/files (displayed in the panel) and the corresponding folders/files in the source control repository. For an individually-selected library, this is achieved using the **Refresh** command, available from the right-click menu in the **All Files** region of the panel. To quickly refresh the status of an entire folder, use the **Ctrl+F5** keyboard shortcut.

---

**Refreshing the status of the source libraries becomes even more important in a multi-librarian environment, where several people may be responsible for maintaining the repository of source symbol and model files.**

---

Status entries pertinent to working with the local libraries of symbols and models are:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>No modification</td>
<td>The local copy of the library is identical to that of the repository. You are free to open the file and make modifications.</td>
</tr>
<tr>
<td>🚨</td>
<td>Out of date</td>
<td>The local copy of the file (in the working folder) is older than its counterpart in the repository, and is therefore out of date. You will need to get the latest version of the file from the repository - updating your own local working copy - before you can modify and commit your own changes. Use the <strong>Update</strong> command (on the right-click context menu) to retrieve the latest file from the repository.</td>
</tr>
<tr>
<td>🔒</td>
<td>Modified</td>
<td>The local copy of the file has been modified in Altium Designer and saved to the working folder. To push your local changes into the repository, simply right-click on the file's entry and choose the <strong>Commit</strong> command from the context menu. Once checked in to the repository, the status will return to <strong>No modification</strong>.</td>
</tr>
<tr>
<td>⚠️</td>
<td>Conflict</td>
<td>The file has been committed by another Altium Designer user before you have committed your own edited and saved version of that file. Use the <strong>Update</strong> or <strong>Resolve</strong> command to determine which version of the file will become the latest revision in the repository.</td>
</tr>
<tr>
<td>⚠️</td>
<td>Locked</td>
<td>The file has been locked by yourself or by another user. A locked file cannot be updated to a new revision in the repository by another user (unless it is forced to be unlocked). This state can be associated with other icons, such as those for Modified 🌐 or No Modification 🌐, when these status conditions also apply.</td>
</tr>
</tbody>
</table>

---

_for more information on working with version control, see [Using Version Control](#).

---

**Physical Comparison of Revisions**

Physical comparison can be performed between any two revisions of the same symbol or model library. This allows you to clearly see, in a visual manner, the changes that have been made - between consecutive or non-consecutive revisions of the file.

Within the **Storage Manager panel**, comparison is performed from the region at the bottom of the panel. It can be easier to switch from the default **Combined View**, to the **Classic View** (by using the **Switch to Classic View** command from the region's right-click context menu), in order to separate the repository and local versioning lists.
The Storage Manager panel presents two views for the history of a selected file. The default is the Combined View, showing the timeline for the file. Hover over the image to see the Classic View, which presents separate regions for the VCS Revisions and Local History. Toggle viewing mode from the right-click menu.

Comparison can be made on one of two levels:

- At the local level - comparing two versions of the document stored within the local working folder. Typically the current version is compared with a local backup (stored in a \History folder, located in the same directory as the source library). Simply select the two versions of the library you wish to compare, in the Local History region of the panel, right-click, and choose Compare.

- At the repository level - comparing two revisions of the document stored within the repository itself. Simply select the two revisions of the library you wish to compare, in the VCS Revisions region of the panel, right-click, and choose Compare.

Once the Compare command is invoked, the two selected document revisions will be opened side-by-side in the workspace, the graphical comparison is made, and any differences detected between the two will be listed in the Differences panel.
Making the Library Available

Like any other library, a standard (or version-controlled) database library is made available through the Libraries panel by adding it to the Available Libraries list. From the Libraries panel, click the Libraries button to access the Available Libraries dialog. The DbLib/SVNDbLib can be added as a project library, or as part of the installed libraries - making it available to all projects and not just the active one. Alternatively, you can specify a search path to the folder in which the DbLib/SVNDbLib file resides.
Making a standard database library (DbLib) available for use by adding it as an installed library. Hover over the image to see the equivalent for a version-controlled database library (SVNDbLib).

The Libraries panel now becomes a 'browsing window' into the linked external database, and the table(s) of components it contains.

**Browsing the Database Components**

Once the DbLib/SVNDbLib has been added to the Available Libraries list, the components in the database will become available for browsing in the Libraries panel. Although only a single DbLib/SVNDbLib file is added, each table in the linked database will present as if it were a distinct library.

Remember that if the target database has been created using multiple Excel spreadsheet files (*.xls), there is a limit of 64 sheets that can connect, due to ODBC driver limitations.

The top drop-down list in the panel will populate with entries of the form:

- `<LibraryName>.DbLib - <TableName>` or `<LibraryName>.SVNDbLib - <TableName>` --where your database has tables
- `<LibraryName>.DbLib - <SheetName>$` or `<LibraryName>.SVNDbLib - <SheetName>$` ----where your database is comprised of sheets in an Excel spreadsheet.
Each component entry in the panel corresponds to a record in that particular table of the database. In fact when browsing loaded database libraries, the Libraries panel behaves like a direct database browser. Symbol and model information is populated from the underlying symbol and model libraries pointed to by the relevant fields of the database (and in conjunction with search paths/locations defined).

By default, only the Part Number and Library Ref fields are displayed in the component listing.
region of the panel. To 'expose' other fields in the database table, simply right-click in the region and choose Select Columns from the context menu. This will give you access to the Select Parameter Columns dialog, from where you can enable the display of any of the additional fields in that table.

**Searching for a Component**

The external database that you have connected to may include a number of tables, each with a sizeable amount of component records. Being able to place directly from a database is one thing, locating the specific component that you want to place is quite another. The latter is made efficiently manageable through the search facility of the Libraries panel.

Click the Search button at the top of the panel to open the Libraries Search dialog, from where you can conduct a search of database components in a single table of an installed DbLib/SVNDbLib file. To enable searching of database components, set the Search Type field to Database Components and select the required Table. Note that the available Fields, listed in the drop-down above, will change to reflect the column headings available in the chosen table. The Scope and Path regions of the dialog become grayed-out, as their options are not relevant when searching database libraries.

When searching database libraries, the search facility essentially offers two 'levels' of searching - Simple and SQL-based querying.
**Simple Search**

This is a 'coarse' search that will return all database components found in the specified table. Simply click the **Search** button, at the bottom-left of the Libraries Search dialog, to begin the search. The dialog will close and the results of the search will be listed back in the Libraries panel. After performing a search, the libraries drop-down list will contain the entry Query Results - <TableName> (n), where n is the number of components returned.

The query results will only be displayed in the Libraries panel if the Components browse mode option for the panel is enabled. Click the button to the right of the library selector to access a pop-up presenting the various browse modes.
SQL Query Search

To perform an SQL query search, click the >>Advanced link in the Libraries Search dialog. The dialog will change, to look like that illustrated in the following image. This level of searching offers a ‘fine’ search based on a specified query expression, where you can enter an SQL query directly into the query editor section of the dialog.

![Libraries Search dialog](image)

Hover over the image to see how the dialog changes when you click Advanced, giving you access to a more powerful SQL-based query search. Click Simple to come back to the simple-style search.

Localized Caching (SVNDbLib Only)

When an SVNDbLib file is added to the Available Libraries list, the symbol and model libraries in the repository are cached in a temporary folder, local to the Designer's computer. When an action in the software calls for the use of a schematic symbol, or model, that symbol/model is retrieved from the local cache.

Automatic Cache Update

The libraries in the cache will be automatically updated with the latest from the source control repository, when performing any of the following actions within the software:

- Placing a component from an installed SVNDbLib file (from the Libraries panel).
- Performing a design transfer (i.e. updating the PCB).
- Using the Update From Libraries command in the Schematic Editor.
- Using the Update From PCB Libraries command in the PCB Editor.

Although the update is automatic, it can be controlled by the use of an option available on the Data Management - SVN Libraries page of the Preferences dialog, highlighted in the following image.
Setting a minimum update time threshold for the local cache. This option essentially provides an 'update timer' for the local cache. If the specified time has elapsed since the last time the cache was updated, then the cache will be updated automatically when one of the listed actions is performed. If an action is performed within this time, the current content of the cache will be used, which may be out of date compared to the content in the repository.

**Manually Forcing an Update of the Cache**

There may be times when you wish to force an update of the local cache, to ensure you have the most up-to-date libraries that are currently stored in the source control repository. For example, when gearing up to transfer the design from the schematic to the PCB.

You can force an update of the cache, manually, using one of the following methods:

- Using one of the **Refresh** commands available from a right-click menu in the Libraries panel.
- Uninstalling and re-installing the SVNDbLib file to the Available Libraries list.

Whenever you restart Altium Designer, the local cache will be automatically refreshed with the latest libraries stored in the SVN repository.
Placing a Component

Placement is performed as it is for any other library loaded and available in the Libraries panel. With the target schematic sheet open as the active document, simply select the component you want to place from the panel and either click the Place button, or drag-and-drop the component directly onto the sheet. The component is built on-the-fly, with the following actions taking place:

- The schematic symbol specified in the database record is retrieved from the relevant schematic component library.
- Any referenced footprint model(s) are linked in (and any referenced PCB3D and simulation models, if using a DbLib).
- The design parameters specified in the DbLib/SVNDbLib file are added (in accordance with the associated Add To Design update setting) to the component.

After placement, double-click on the component on the schematic to access its properties dialog. You will see the design parameters added to the Parameters region of the dialog, the linked models added to the Models region and another region - Link to Library Component. This region provides the following information:

- The name of the parent DbLib/SVNDbLib file.
- The specific database table in which the component resides.
- The value for Design Item ID (Component Name / Part Number). This is the chosen key field defined in the Field Settings region of the DbLib/SVNDbLib file, and is typically the part number.
Information added to a component placed from a database, linked-to using a DbLib file. Hover over the image to see the example when placed from a database that is linked-to using an SVNDbLib file.

You can change the placed component to another from the same table by clicking the **Choose** button. This will open the **Browse Libraries dialog**, initially listing all components in the same table. In fact, you can change to a component in a different table of the same DbLib/SVNDbLib file, or browse to one in a different DbLib/SVNDbLib file altogether. The region will update accordingly with the new information for the chosen database component.
Ensuring Synchronicity

After placement, the chosen key field parameter is used to ensure that the placed component on the schematic, retains its link to the corresponding record for that component in the external database. This means that at any stage in the future, changes to parameter and model information in the database can be easily passed back to the placed component, synchronizing the two.

If you simply want to update parameter information, use the [Update Parameters From Database command](#), available from the Schematic Editor's main [Tools](#) menu.

To perform a full update, including parameters, model and graphical attributes of schematic symbols, use the [Update From Libraries command](#) (also available from the Schematic Editor's main [Tools](#) menu).

In the PCB Editor, use the [Tools » Update From PCB Libraries command](#) to update placed footprints with the latest information stored in the source libraries.

### Dual Synchronization - DbLib/SVNDbLib and DBLink

You may have an existing design project where the majority of the placed parts have been linked to an external database using a DbLink file. Design changes may result in additional circuitry, the components for which might be placed using the [Database Library](#) feature. The associated DbLib/SVNDbLib file could, quite conceivably, point to a different external database.

When using the [Update Parameters From Database command](#), all linked parameters for placed components will be queried - across all linked databases, irrespective of the linking method used - and detected differences for those parameters displayed in the [Select Parameter Changes dialog](#).

If the same database field has been used for matching in both the DbLink and DbLib/SVNDbLib files, the database linked by the DbLink file will be searched first for a match, followed by the database linked to by the DbLib/SVNDbLib file. If the component is present in both databases, you could quite possibly match and update from the wrong external record.

### Adding Database Information Directly to a BOM

*Related page: Bill of Materials (BOM)*

Source information for a Bill of Materials (BOM) has, in the past, been taken from the parameter information of the placed components for the design. But that can lead to a lot of information attached to a schematic that is only ever used for the BOM. If your placed Altium Designer components are linked to an external database using a DbLib/SVNDbLib file, the BOM Generator is able to extract any other record information that has not been added to the components as design parameters.
Parameters can be included from an external database, irrespective of the method employed to link to that database - DbLink, DbLib, or SVNDbLib.

When configuring the Bill of Materials report using the Report Manager dialog, simply enable the Include Parameters from Database option. This option will only be available if one or more components in your design are linked to an external database. In the parameter listing, the icon is used to distinguish a parameter that exists for one or more placed components in a linked external database.

For a design that includes Variants, the parameters used to query the database are varied in the Variant Management dialog. Since those parameter values only exist in the Variant Management dialog, they are only used to query and match records in the database when a BOM is being configured for generation. To do this, ensure the sub-option - Include in Component Variations - is enabled.

Source URL: https://www.altium.com/documentation/ru/display/ADES/((Working+with+Database+Libraries))_AD