Database Overview

Main Features

Built-in Database
Slick! allows you to attach a database record to any file in your system. You are not limited to AutoCAD files!

Query Capabilities
You may query the database for information based on a set of conditions you specify.

Set Display and Report Layouts
You may control the way database information is displayed and reported.

Browse Feature
You may browse thru the contents of the database.

Import/Export Data:
Data may be imported to or exported from other databases.

Bidirectional Attribute Exchange:
Attribute data from AutoCAD drawings may be imported directly into Slick!'s database. Conversely, Slick! database data may be used to update attribute data in your AutoCAD drawings.

Year 2000 compliant
Date fields now require century.

Usage

Frequently, the AutoCAD user requires additional information to be attached to a drawing. These include data such as drawing number, title, author's name, drawing revision, and so forth. Slick! allows you to create and customize a database based on your requirements. A database file called *slickdb.dbf* will be created and maintained by Slick.

Basic Database Concepts

✍ **The Slick! database file *slickdb.dbf* is a separate file from your drawing or graphics files.**

✍ **It is a single file consisting of several records.**

✍ **You may create a record for any file you wish, one record per file.**

✍ **Each record is comprised of several fields. A field contains an item of information that relates to the drawing. For example, drawing number or title.**

✍ **Each field has a name, data type, and length which are defined by the database structure.**

✍ **Slick! requires four separate fields. One each for the DRAWING NAME, PATH, EXTENSION and ARCHIVED (which contains the drive letter where the file is located). Together these fields constitute the key which Slick! uses to associate a database record to the file.**

slickdb.dbf file and SLICKDBF environment variable

If you did not specify a User Directory during database configuration, Slick! will create slickdb.dbf in the directory *slickdb.dir* at the root level of your harddrive. If you have files across multiple harddisk drives, Slick! will create this directory in each drive!

You may however specify a different directory for *slickdb.dbf* by setting the environment variable SLICKDBF. This is a powerful feature since you can now specify multiple databases using identical or different structures and still be able to choose which one to use during a Slick! session. For more information, see **Multiple Databases** below.

When you select a file to view, Slick! checks if SLICKDBF has been set.

· If SLICKDBF is **not** set, it looks for *slickdb.dbf* in *slickdb.dir* in the root level of the drive containing the file being viewed.

· If SLICKDBF has been set, it looks for *slickdb.dbf* in the directory specified by SLICKDBF. You must create this directory before running Slick! Otherwise, you will receive the message:

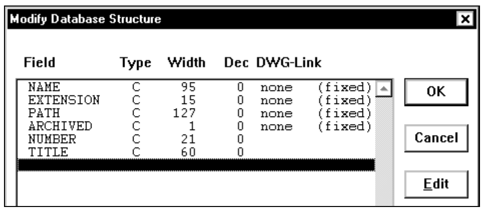
```
Invalid SLICKDBF path:  <your pathname>
Using \SLICKDB.DIR
    press a key
```

If *slickdb.dbf* does not exist in the *slickdb.dir* or the directory specified by SLICKDBF, Slick! displays the message:

No database created for selected drive (directory)? Create one? Enter **Y** or **N**.

IF YOU HAVE CONFIGURED SLICK! TO USE "USER DIRECTORY", IT LOOKS FOR slickdb.dbf IN THIS DIRECTORY. IT COMPLETELY DISREGARDS THE CONTENTS OF SLICKDBF!

Default Database Structure



```
Field        Type  Width  Dec  DWG-Link
NAME         C       95   0    none    (fixed)
EXTENSION    C       15   0    none    (fixed)
PATH         C      127   0    none    (fixed)
ARCHIVED     C        1   0    none    (fixed)
NUMBER       C       21   0
TITLE        C       60   0
```

The initial database created by Slick! contains six fields as shown below. Their definitions are fixed and may not be modified. The first four fields: *Name, Extension, Path, and Archived* are internal to Slick! The last two fields: *Number, and Title* may be used to enter the drawing number and a descriptive title. The information under the column DWG-Link relates to block attribute tag names. Drawing Links are discussed in the section on *Bidirectional Attribute exchange.*

Field Name	Type	Width	Decimal	DWG-Link
NAME	C	95	0	none
EXTENSION	C	15	0	none
PATH	C	127	0	none
ARCHIVED	C	1	0	none

NUMBER	C	21	0	none
TITLE	C	60	0	none

Name	- corresponds to the file name
Extension	- extension;
Path	- full path name;
Archived	- drive letter where the file is located;

Number- May be used to contain the drawing number;
This should be a unique field, that is, the user should not assign the same number to different drawings in the database file. Although Slick! allows duplicate numbers, this is not recommended! See Configure Database in Installation section.

Title - May be used to contain a descriptive title for the drawing;

The user has the option of modifying the database structure to add up to thirty four (34) additional fields as needed.

You may attach a database record to any file on your harddisk. For example: Lotus or Quattro Pro spreadsheet files, WordPerfect documents, Presentation files, scanned files, and others.

Frequently asked database questions

Q. How do I copy the structure of an existing slickdb.dbf to another drive or directory without having to reenter all the field names?
A. Use the Slick! Copy command to copy an existing file to the target drive or directory and then delete the file afterwards. The copy command will automatically create slickdb.dbf in the target drive or directory with the same structure as the existing one.

Q. We have an existing dBASE DBF file already and would like to use Slick! to browse through the records. What do I do?
A. Modify the structure of your existing DBF file in dBASE as follows:
?? The first six fields must be those required by Slick! and in their proper order and definition;
?? The PATH, and NAME fields must have valid data.
?? The ARCHIVED field must contain the drive letter where the file resides.
?? The file must be named SLICKDB.DBF and placed in the User Directory, SLICKDB.DIR directory or the directory set by

SLICKDBF;
- ?? Memo fields are not supported;
- ?? The EXT, NUMBER, and TITLE fields may be blank.

Q. Does Slick! modify my AutoCAD drawing to add database information?
A. No. The database file slickdb.dbf is a separate file from the drawings files. The database file is a collection of records. Each record contains the path and extension of the drawing it is associated with.

Q. If I copy a drawing from one drive to another using DOS, will the database information be automatically copied to the database file in the target drive?
A. Absolutely not. You must use the Slick! copy command to copy files to other drives if you wish to copy the database information.

For answers to more database-related questions go to the "Frequently asked questions" chapter at the end of the manual.

Multiple Databases

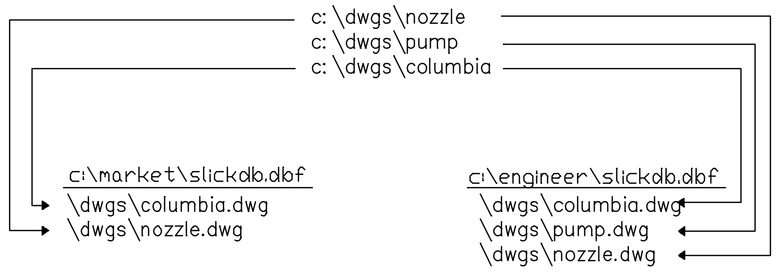
Most users will only need one centralized database to contain information on all their files. The location of the database would be set in the User Directory you specify at configuration.

If necessary you can define multiple databases in Slick! but Slick! can access only one database at a time. Inform Slick! of the database to use by:
- ?? reconfiguring Slick! to use a different database by changing the User Directory or
- ?? setting the environment variable SLICKDBF to point to another directory before entering Windows

If User Directory is set, it overrides any setting of SLICKDBF.

For instance, the marketing department in your company might require different information fields from the engineering department. In this case, you may define a directory called *c:\market\* for marketing and *c:\engineer\* for engineering. .

In the example below, you have three drawings in path *c:\dwgs* and two databases keeping track of these drawings. The database file *c:\market\slickdb.dbf* contains a record for columbia and a record for nozzle. The database file *c:\engineer\slickdb.dbf* contains a record for columbia, pump, and nozzle.



Multiple Databases Example

Database compatibility with older versions

Databases created with versions of Slick! v4.0 and older are not compatible with Slick! v8.0.

When you first use Slick! v8.0, it will automatically convert your database to the new format. See Installation chapter.

The old database files are:	slickdb.dbf and slickdb3.cdx.
The new converted files are:	slickdb4.dbf and slickdb4.cdx

If your company uses multiple copies of Slick! for Windows and access information from the same database, each user should upgrade to the newer version to maintain compatibility.

Since V8.0 maintains drive information, it is now ideal for you to maintain one central database for all your drives. See database configuration and usage of User Directory.

Database Modify Structure

Database ? DB Structure command

This section discusses how you can modify the default Slick! database structure to specify additional fields as needed.

Important Precautions

? Backup all your database files (.dbf) in slickdb.dir or your user directory before modifying the database structure as a precaution against the risk of losing valuable data!

? You should not change a field name and its width or type at the same time! If you do, Slick! will not be able to append data from the old field and your new field will be blank! First change the name, save the new structure, and then Edit the database structure again to change field width or type!

? You should not insert or delete fields from the database and change field names at the same time! Slick! appends data from the old file by using the field position in the file. Inserting or deleting fields as well as changing field names will cause loss of data! First change the name, save the new structure, and then Edit the database structure again to insert or delete fields!

Database Structure

Each time you enter database information for a file, a record gets added to the *slickdb.dbf* file. The record contains the minimum six fields that Slick! requires. You may define up to thirty four (34) fields in addition to these minimum fields. A field is defined by its field name, type, width, and the number of decimal positions if the type is numeric, and an *optional* drawing link (DWG-Link).

In general, Slick! maintains one database directory *slickdb.dir* per harddisk drive. Each directory contains a database file *slickdb.dbf*. Database files .dbf(s) may have different database structures. The Edit command modifies the database structure of the currently selected drive. If you have multiple drives, be aware that the changes being made affect only the database structure of the current drive!

If you have set the environment variable **SLICKDBF**, Slick! will modify the structure of the *slickdb.dbf* file residing in the directory specified by **SLICKDBF**.

Database Field Definition

Field Name

The field name may be up to 10 characters long, and may consist of letters, numbers, and the underscore character. The field name cannot contain embedded blank characters and the first character of the field name must be a letter.

Type

The valid field types are:

C Character field; maximum width is 255.
N Numeric field; maximum width is 17 including the decimal point.
L Logical field; predefined width is 1; acceptable values are **T, F, Y, N**
D Date field; predefined width is 8; only valid dates are acceptable. Slick! displays and accepts date fields in MM/DD/YY format. Internally however, Slick! stores date fields in YYYYMMDD format. This is of importance only to those who would be importing data into Slick! See Import/Export section.

Width

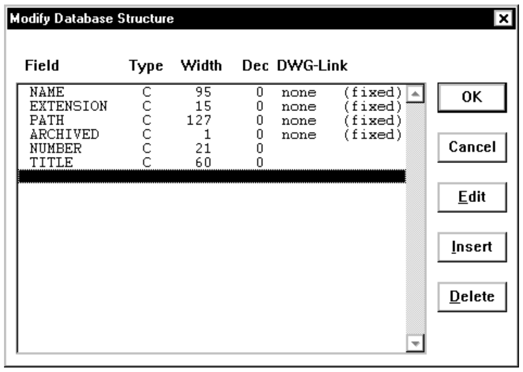
Field width. For numeric fields, width must account for the number of decimal places including the decimal point.

Decimal Places

Number of decimal places for numeric type fields.

DWG-Link (Drawing Link)

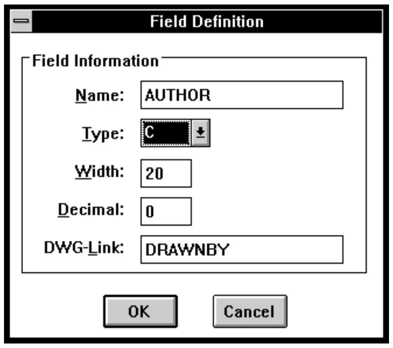
Optional item that you may specify if you wish to use the bidirectional attribute exchange feature of Slick! If used, it must contain either the tag name of a block attribute defined in your AutoCAD drawings or one of the supported AutoCAD variables. See section on Bidirectional Attribute Exchange.



```
Field         Type   Width   Dec  DWG-Link
NAME          C        95     0   none    (fixed)
EXTENSION     C        15     0   none    (fixed)
PATH          C       127     0   none    (fixed)
ARCHIVED      C         1     0   none    (fixed)
NUMBER        C        21     0
TITLE         C        60     0
```

Edit

You must first pick the field to be edited (highlighted with a black horizontal bar), before picking the *Edit* menu. Slick! will then proceed to prompt you for Field Name, Type, Width, Decimal Places, and DWG-Link. Existing values will be shown as defaults. Select OK when finished.



Name: AUTHOR
Type: C
Width: 20
Decimal: 0
DWG-Link: DRAWNBY

You may not change the name, type, width, and decimal positions of Slick!'s fixed default fields. You may however specify the optional DWG-Link to the NUMBER and/or TITLE fields.

Insert

First highlight a position in the structure screen before selecting the Insert command. Slick! will provide a space to insert the new field definition. To add a field definition to the end of the list, select a point after the last field.

Delete

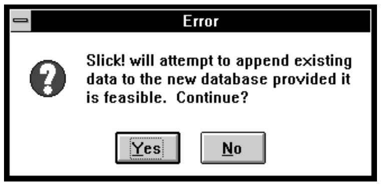
Select a field name and then pick Delete to delete the field. Slick! prompts for confirmation. Select Y or N.

Save Changes and exit

Select OK when finished with your changes to the structure.

? Read Precautions at the beginning of this section before saving your changes!

Slick! will display:



Slick! will attempt to append existing data to the new database provided it is feasible. Continue?

Select *Y* or *N*.

Database Utilities

This section discusses miscellaneous utility functions available from the Database menu.

Delete Record

Marks a record for deletion. The word *Deleted* will appear on the right corner of screen. You may undelete previously deleted records as long as they have not been purged. See *Purge* command below. If the record being deleted has the same **NUMBER** as another record, Slick! displays:

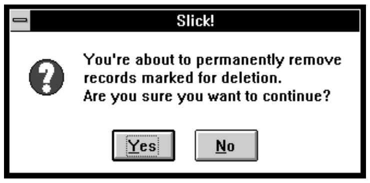
Warning. Another record has the same NUMBER field. Write record anyway? <N> Enter *Y* to continue delete. Slick! displays: *Record deleted*.

Undelete Record

Undeletes a record previously marked for deletion. See *Delete Record* command above.

Purge

Permanently deletes records marked for deletion from the database. Slick! displays:



Slick!

You're about to permanently remove records marked for deletion.
Are you sure you want to continue?

Record Count

Counts the number of records in the database. Slick! displays: *Number of records nnn* Press OK to continue.

Reindex

Reindexes the database. This command is only necessary if Slick!'s data base file has been modified **externally**. See section on *Database Import*. Slick! displays:

Database successfully reindexed.

Add Dummy Record

Adds a dummy record for non-existent records.

You can create a database record which has no association or link to any particular file. This is useful for storing information for non-electronic media such as paper drawings. Slick! stores these records with the false directory path of **"\DUMMY-DBF-RECORDS"**. In what would normally be a filename, Slick! automatically generates a sequential 8-character numeric name with extension of **.DMY**.

Enter information normally as with any other database record. The contents of these records will be displayed as well if you browse thru the database.

Browsing for dummy records

If you wish to browse through the dummy records only, you can build the following query condition:

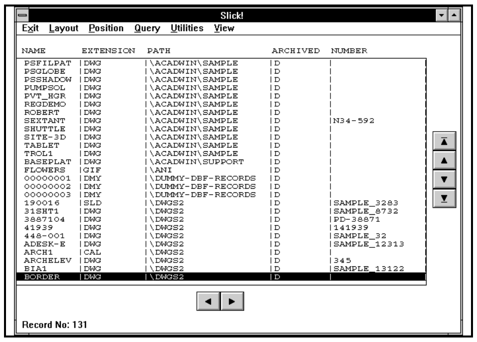
```
PATH = '\DUMMY-DBF-RECORDS'
```

? Note that the directory path should be in all capitalized letters!

Database Browse

Database ? Browse command

Use this command to browse thru the contents of your database. Upon entry into the browse screen, Slick! displays the database contents by field name from left to right.



```
NAME      EXTENSION  PATH                   ARCHIVED  NUMBER
 PSFILPAT |DWG       |\ACADWIN\SAMPLE        |D        |
 PSGLOBE  |DWG       |\ACADWIN\SAMPLE        |D        |
 PSSHADOW |DWG       |\ACADWIN\SAMPLE        |D        |
 PUMPSOL  |DWG       |\ACADWIN\SAMPLE        |D        |
 PVT_HGR  |DWG       |\ACADWIN\SAMPLE        |D        |
 REGDEMO  |DWG       |\ACADWIN\SAMPLE        |D        |
 ROBERT   |DWG       |\ACADWIN\SAMPLE        |D        |
 SEXTANT  |DWG       |\ACADWIN\SAMPLE        |D        |N34-592
 SHUTTLE  |DWG       |\ACADWIN\SAMPLE        |D        |
 SITE-3D  |DWG       |\ACADWIN\SAMPLE        |D        |
 TABLET   |DWG       |\ACADWIN\SAMPLE        |D        |
 TROL1    |DWG       |\ACADWIN\SAMPLE        |D        |
 BASEPLAT |DWG       |\ACADWIN\SUPPORT       |D        |
 FLOWERS  |GIF       |\ANI                   |D        |
 00000001 |DMY       |\DUMMY-DBF-RECORDS     |D        |
 00000002 |DMY       |\DUMMY-DBF-RECORDS     |D        |
 00000003 |DMY       |\DUMMY-DBF-RECORDS     |D        |
 190016   |SLD       |\DWGS2                 |D        |SAMPLE_3283
 31SHT1   |DWG       |\DWGS2                 |D        |SAMPLE_8732
 3887104  |DWG       |\DWGS2                 |D        |PD-38871
 41939    |DWG       |\DWGS2                 |D        |141939
 448-001  |DWG       |\DWGS2                 |D        |SAMPLE_32
 ADESK-E  |DWG       |\DWGS2                 |D        |SAMPLE_12313
 ARCH1    |CAL       |\DWGS2                 |D        |
 ARCHELEV |DWG       |\DWGS2                 |D        |345
 BIA1     |DWG       |\DWGS2                 |D        |SAMPLE_13122
 BORDER   |DWG       |\DWGS2                 |D        |

Record No: 131
```

The archive column indicates the letter of the drive containing the file. Scrolling and paging arrows are provided for controlling the screen display.

◄	Shift screen one field to the left
▼	Scroll up one record
⤓	Page up
▲	Scroll down one record
⤒	Page down
►	Shift screen one field to the right

? The information displayed on the screen is affected by both the *Query* and *Set Layout* commands.

You may do a query from within the browse command or from the database menu in the menu bar. The query command is discussed in detail in the next section. If a query table has been loaded or is being processed, only those records which meet the conditions will be displayed. If no records qualify, the screen will be blank and **Record**

0 will be indicated at the bottom of the screen.

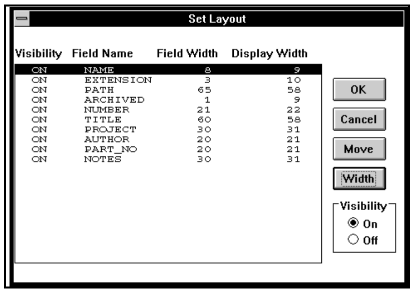
If you have previously set or loaded a display layout, data will be displayed as defined in the layout. *See Database Query, Set Layout, Load Layout, Reset Layout*.

Layout menu

This pull down menu allows you to arrange the layout of information on the browse screen.

Set Layout

This command controls the manner in which database information is displayed on the screen and reported. See Database Reporting section. Normally, Slick! displays field information in the same order and width as defined in the database structure. With this command, the user has the option of changing the order in which fields are displayed, the display field width, and whether specific fields are to be visible or not. Slick! maintains current Layout settings in a temporary buffer area. The user may **save** (see Save Layout) the current settings to a disk file. The saved settings may then be **loaded** (see Load Layout) later as needed without the need for re-entering the settings.



Visibility	Field Name	Field Width	Display Width
ON	NAME	8	9
ON	EXTENSION	3	10
ON	PATH	65	58
ON	ARCHIVED	1	9
ON	NUMBER	21	22
ON	TITLE	60	58
ON	PROJECT	30	31
ON	AUTHOR	20	21
ON	PART_NO	20	21
ON	NOTES	30	31

Visibility: [✓ On] Off

Visibility

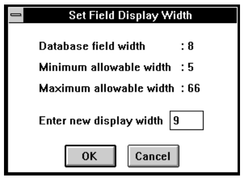
Select the desired field and pick the desired visibility option, ON or OFF. If visibility is set to OFF, the field will not be displayed on the browse screen or your reports.

Move

Use this command to move the position of the field to be displayed relative to other fields. Highlight the field to be moved by picking it and then pick Move from the dialogue box. When Slick! displays *Select the new field position*, pick the new position of the field. The other fields will be moved accordingly.

Width

Use this command to control the display field width. Highlight the desired field by picking it and then pick Width from the screen menu.


Set Field Display Width

Database field width : 8
Minimum allowable width : 5
Maximum allowable width : 66

Enter new display width 9

Enter the new display width and select OK when finished.

Reset Layout

Use this command to reset the display and report layout to default values. This means that all fields become visible, field positions are in the same order as they are defined in the database structure, and the display width of each field is the same as defined in the database structure. Slick! displays:
Layout successfully reset. Pick any point to clear message.

Save Layout

Use this command to save the current layout to disk for subsequent loading. See Load Layout below. It overwrites any previously saved layout. Slick! **does not** support multiple layouts. It is suggested that the user save the most often used layout. Slick! displays: *Data successfully saved.* Pick any point to clear message.

Load Layout

Use this command to load the last saved layout. Slick! displays: *Layout successfully loaded.* Pick any point to clear message. The saved browse layout may be automatically loaded if you configured Slick! to do so. See Configure Database.

Position menu

This pull down menu allows you to quickly move to any record within the database.

Top Record

Goes to the first record in the database.

Bottom Record

Goes to the last record in the database.

Record Number

Prompts the user for a record number and goes to that record.

Skip

Prompts for a specified number of records to skip and goes to that record. Note: Positive number skips forward; negative number skips backward.

Utilities menu

This pull down menu allows you to delete and undelete database records and to set sort on or off.

Delete Record

Use this command to mark a record for deletion. Deleted records may be permanently removed from the database later with the *Purge* command from the main database menu. If the record being deleted has the same **NUMBER** as another record, Slick! displays: *Warning. Another record has the same NUMBER field. Write record anyway? <N>* Enter *Y* to continue delete. Slick! displays: *Record deleted*. Pick any point to clear message.

Undelete Record

Use this command to undelete a record previously marked for deletion. Slick! displays: *Record undeleted*. Pick any point to clear message.

Set sort on

This command sorts the database records on the currently highlighted field.

? Database reports will reflect the current sort order of database records. See the Database Reporting section.

Reset sort

This command resets the sort sequence by record number.

View

Use this command to view the selected file. If the file has previously been deleted, Slick! displays: *Unable to open <file name>*.

You can only view the drawing associated with the database record if the drawing exists in the currently drive!

Note: It is possible to have database records for drawings residing in drives different from the drive in which *slickdb.dbf* resides.

Exit

Exits the browse screen and returns to the main display.

Database Query

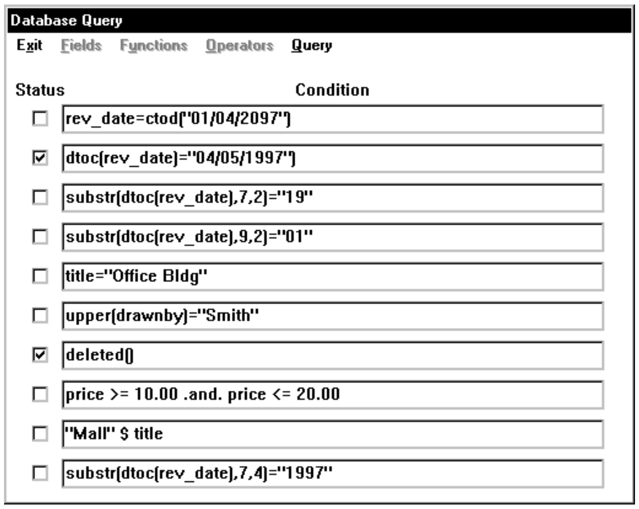
Database ? Query command

Query allows you to build query conditions against the database. This is useful if you wish to see information only on files that meet certain conditions. For example, all drawings belonging to a specific project. Query conditions affect both the display and reporting of data. See *Browse*, and *Reporting* sections.

You may invoke the Query command from the Database menu or from the Database ? Browse menu.

Upon entry, Slick! displays the Query screen. Initially, it will be empty. The following shows a sample query screen in which condition statements have been entered.

Sample Query Screen

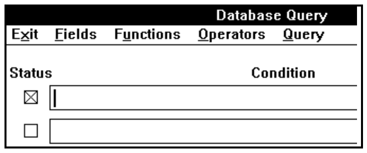


Status	Condition
☐	rev_date=ctod("01/04/2097")
☑	dtoc(rev_date)="04/05/1997")
☐	substr(dtoc(rev_date),7,2)="19"
☐	substr(dtoc(rev_date),9,2)="01"
☐	title="Office Bldg"
☐	upper(drawnby)="Smith"
☑	deleted()
☐	price >= 10.00 .and. price <= 20.00
☐	"Mall" $ title
☐	substr(dtoc(rev_date),7,4)="1997"

You may use the *Fields, Functions, and Operators* pull-down menus to assist you in building a set of conditions which will subsequently be tested by Slick! Initially, Slick! displays ten blank query lines. The entire set is hereinafter called the *Query Table*.

Condition Status

You may set the status of a condition ON or OFF by clicking on the small box to the left of the condition box. The condition status is ON if the box contains an **X**. The status of each condition line is initially set to OFF.



Condition status is ON if box is marked by an **X**. Condition statements in the query table are held by Slick! in a temporary buffer area. The user has the option of saving the query table to a disk file. The saved query table may then be loaded later as needed without the need for you to re-enter the condition statements.

Syntax Checking

? ? Slick! validates the query conditions entered by the user for syntax errors and data type inconsistencies. If Slick! detects an error, an appropriate error message will be displayed. If you cannot correct the error and simply wish to exit, clear the condition line containing the error and exit normally. Pressing escape will not work!

Building condition statements

To enter a query condition, simply pick the desired condition box and start typing the condition statement.

You may set the status ON or OFF at any time. If the condition is OFF, Slick does not test the condition. You may turn any or all conditions ON or OFF when browsing or reporting data from the data base.

? If a query is being processed, Slick! will only display those records which meet all the conditions that are turned ON!

You may use the **Fields, Functions, and Operators** menus to assist you in building the condition.

Note: These menus are available only if you have selected a condition box. Otherwise, they are dimmed.

Fields menu

This command displays a list of field names as defined in the database structure. Double-click on the desired field and the field name will be inserted automatically in the current condition box. Press the space bar to provide a space between the field names and operators.

Operators menu

This command displays a list of operators. Operators are symbols which link fields, constants, and functions so that Slick! can evaluate the expression or condition as one unit. Double-click on the desired operator and the operator will automatically be inserted on the condition line.

Slick! provides four types of operators: mathematical, relational, logical, and string.

Mathematical Operators

Mathematical operators generate numeric results.

+	Addition/Unary Positive
-	Subtraction/Unary Negative
*	Multiplication
/	Division
**	Exponentiation
()	Parentheses for grouping

Relational Operators

Relational operators generate logical results, that is (.T.) or (.F.). You can use relational operators with character, numeric, or date expressions. Both expressions you use in a relational operation must be of the same type.

<	Less than
>	Greater than
=	Equal to
<> or #	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
$	Contain or substring comparison.

Contain ($) operator

This operator is a powerful one and is explained further. You can use this operator to test if one string is identical to or contained within a second string. That is, if A and B are both character strings, A$B returns a logical true if A is either identical to B or is contained within B.

For example, if you are searching for all files with the word "defense" in the TITLE field, the condition line should read:

```
"defense" $ TITLE
```

Logical Operators

Logical operators obtain a logical result from comparing two expressions.

.AND.	Logical and
.OR.	Logical or
.NOT.	Logical not
()	Parentheses for grouping
=	Equal to
<> or #	Not equal to

String Operators

String operators concatenate two or more character strings into a single character string.

+	Trailing spaces between the strings are left intact when the strings are joined
-	Trailing spaces of the string preceding the operator are moved to the end of the last string
()	Parentheses for grouping

For example, if string A contains *"ABC   "* and string B contains *"DEF"*:

the expression A + B results in *"ABC   DEF"*
while A - B results in *"ABCDEF   "*

Precedence of Operators

Operators have a precedence which specifies operator evaluation order. The precedence of each operator is specified in the following tables. The higher the precedence, the earlier the operation will be performed. Divide has a precedence of 6 and Plus has a precedence of 5 which means Divide is evaluated before Plus. Consequently 1 + 4 / 2 returns the value 3. Evaluation order can be made explicit by using parentheses. For example, 1 + 2 * 3 returns 7 whereas (1 + 2) * 3 returns 9.

In general, when several of the four types of operators are used in the same expression, the precedence levels are:

- mathematical or string (highest precedence)
- relational
- logical (lowest precedence)

All operations at the same precedence level are performed in order from left to right. Use parentheses to override the order in which operations are performed. Operations within the inner-nested parentheses are performed first.

	Operator	**Description**	**Precedence**
Logical Operators	.NOT.	Logical not	1
	.OR.	Logical or	2
	.AND.	Logical and	3
Relational Operators	=	Equal to	4
	<>	Not equal to	4
	<	Less than	4
	>	Greater than	4
	<=	Less than or equal to	4
	>=	Greater than or equal to	4
	$	Contain	4
Math Operators	+	Add	5
	-	Subtract	5
	*	Multiply	6
	/	Divide	6
	**	Exponentiation	7

Functions menu

This command displays a list of functions available for data conversion. Double-click on the desired function and the function will automatically be inserted in the condition box.

A function can be used as an expression or as part of an expression. Like operators, constants, and fields, functions return a value. Functions are always followed by left and right parenthesis. Values (parameters) may be included inside the parentheses. Values for character strings must be enclosed in quotes.

When you defined the structure of your database, you specified the field type for each field. The field type may be character, numeric, logical, or date.

Field types must match

Before you can compare the contents of two fields, they must be of the same type. If they are different, you must you a function to convert the field type of one to that of the other!

Comparing Numeric fields to Character fields

You must use either:
- the VAL() function to convert the character string to a numeric value before comparing it with the numeric field, or
- the STR() function to convert the numeric value to a character string before comparing it with the character field.

Comparing Date fields to Character fields

All date fields now have a width of 10 to accommodate the new century.
The format is MM/DD/CCYY where CC is the century.
You must use either:
- the CTOD() function to convert the character string to a date value before comparing it with the date field, or
- the DTOC() function to convert the date field to a character string before comparing it with the character field.

Functions Defined

CTOD()

Character String to Date Variable Conversion

Syntax: *CTOD("MM/DD/CCYY")*
Usage: To convert a character string to a date type variable and to compare the result to a date field in the data base.
Example: You have defined a date type variable called REV_DATE in the Slick! database and you wish to test if it contains the date February 26, 1989. Your query condition line should then read:

```
REV_DATE = CTOD("02/26/89")
```

DTOC()

Date Variable to Character String Conversion

Syntax: *DTOC(Date_Value)*
Usage: This is the opposite of the CTOD function above. It converts a date variable to a character string in the form

"MM/DD/CCYY" so that you can compare it to another character string or character field. The parameter Date_Value must be defined as a date variable field type in the database.
Example: You have defined a date type variable called REV_DATE in the Slick! database and you wish to test if it contains the date February 26, 1989. Your query condition line should then read:

```
DTOC(REV_DATE) = "02/26/1989"
```

DELETED()

Function to test if current record is marked for deletion. Note: If you have just previously purged the database of records marked for deletion, no records will qualify for this test! See the Purge command in Database Utilities.

Syntax: *DELETED()*
Usage: This function may be used as a query condition to browse for database records that have been marked for deletion. Conversely, it can be used to browse for records that have *not* been marked for deletion.

Example: The query condition line may simply contain:

`DELETED()`	to test for deleted records or
`.NOT. DELETED()`	to test for not deleted records

IIF()

This function stands for an immediate IF and is a shortcut to the IF...ENDIF programming construct.
Syntax: IIF(<condition>, <expression 1>, <expression 2>)
Usage: If the logical expression specified by <condition> is true, the function returns the result of the first expression; otherwise, it returns the result of the second expression.
Example: In the following example, your database contains a field called LAST_NAME and you will be creating a new character string called NAME depending on the contents of field SEX:

```
NAME =IIF(SEX="F", "MS. ", "MR. ") +
LAST_NAME
```

RECNO()

Returns the record number of the current record.

Syntax: *RECNO()*
Usage: Displays only those records whose record numbers are within a specified range.
Example: If you wish to display only those records whose record numbers are 10 thru 20, you will build the following query condition line:

```
RECNO( ) > 9 .AND. RECNO( ) < 21
```

STR()

Converts a number into a character string.
Syntax: *STR(Number, Length, Decimals)*
Usage: The string function converts the value of a numeric field into a character value. *Length* is the number of characters in the new string, including the decimal point. *Decimals* is the number of decimal places desired. If the number is too big for the alloted space, asterisks (*'s) will be filled in instead.
Example: Assuming *COST* is a numeric field containing the value 25: `STR(COST,5,2)` returns the character string `"25.00"`. `STR(COST,4,2)` returns "****" since not enough space is alloted for the new string.

SUBSTR()

Returns a specified number of characters from a character expression.

Syntax: *SUBSTR(Character Expr., Starting Position, Number of Characters)*
Usage: *SUBSTR* is useful in extracting embedded characters from a character string.
Example: The following condition will display only those drawings with the letters "XYZ" starting in position five of the TITLE field.

```
SUBSTR(TITLE,5,3) = "XYZ"
```

UPPER()

Converts a character string to uppercase letters.

Syntax: *UPPER(Char_Value)*
Usage: Converts information in a character field to uppercase letters. This function is useful when you wish to search for a specific string value in a field but are not sure if the field contents were entered in lower or uppercase letters.
Example: You want to set a query condition to find all the drawings created by a draftsperson named Robert in database field AUTHOR. Since the name might have been entered as "Robert", "ROBERT" or a mixture of lowercase and uppercase letters, the query line should read as follows:

```
UPPER(AUTHOR) = "ROBERT"
```

VAL()

Converts a character string to a numeric value.

Syntax: *VAL(Char_Value)*
Usage: A character field in the database needs to be converted to a

numeric value for comparison purposes.
Example: You have a field called COST in your database which was defined as a character field type and you wish to query for all records whose cost is equal to 150.00. The query line should read as follows:

```
VAL(COST)= 150.00
```

Query menu

Do Query

This command will process any query condition that is turned on and exit to the main menu. As opposed to the browse command above which enters browse mode immediately, this command allows you the option of picking Report from the Database menu to generate reports. You may of course still enter browse mode from the main menu.

Reset Query

This resets all query conditions to none. That is all database records will be available for viewing. After execution, Slick! will display the message:
Query reset successful.

Save Query

This saves all the query conditions in the current query table to a file. The user may subsequently load this table repeatedly for query purposes. If a query table has previously been saved, Slick! will prompt: *Overwrite previously saved information?* Respond with *Y* to save the new query table.

Load Query

This loads the query table previously saved by Save Query. The condition statements therein will now be in effect. Slick! will display the message:
Query successfully loaded.

The query conditions can be automatically loaded if you have configured Slick! to do so. See Configure Database.

Exit menu

Select to exit query. You will return either to the main screen or to the Browse menu depending on where you entered Query.

Any changes you have made to the query conditions *will not* be placed into effect. However, changes made to the query table are kept in the buffer area. The previous query conditions are still in effect.

Query Example

The following example illustrates a simple query application.   Let us assume that you wish to locate all files whose TITLE field contains the characters ASSY.

· Modify the contents of the TITLE field for two or three sample files to ASSY1, ASSY2, ASSY3, etc.
· Pick Database ?   Query from the menu bar.
· The query screen will appear with ten blank condition statement lines.
· The status of all the lines are initially *OFF*.
· Select the first condition box.
· Pick *Fields* from the menu bar to display a list of field names.
· Double-click on TITLE from the field list.
· TITLE will appear on the condition line.
· Press the space bar to enter a space.
· Pick *Operators* from the menu bar to display a list of operators.
· Double-click on = from the list.
· Press the space bar to enter a space.
· Type ASSY in the condition box.
· The condition line should now show:
  TITLE = "ASSY"
· Pick on the status box beside this condition line to set status ON.
· Pick *Exit* from the menu bar.
· Pick *Do Query* from the submenu.
· Slick! returns to the main menu.
· Pick *Browse* from the Database menu.

Notice now that only those drawings whose TITLE begins with *ASSY* are shown on the screen!

Query Conditions Notes

The *Fields* and *Operators* and *Functions* menus are simply tools to assist you in building the condition line. It is not necessary to use them. You can simply type the complete condition statement using the keyboard.

Case Sensitivity

All condition statements are case sensitive. Uppercase strings compared with lowercase strings will not return true. For example, *"PROJECT" = "Project"* will return false.

String comparisons

Comparisons between character strings begin with the left character in each string and continue character by character to the end of the string to the <u>right</u> of the relational operator. If the two strings are the same up to that point, the result of the comparison is true. For example, if you are comparing string **A** to string **B**, make sure that the number of characters in string **A** is equal or greater than string **B**. Otherwise, you will never get a valid result! ? *Exception is the contain function (see discussion below).*

$ Contain substring comparison

This is a powerful function to test if a field contains a character string. **The string must be to the left of the $ operator.** For example, if you wish to test if the field **TITLE** contains the word **"test"**, use the following statement: **"test" $ TITLE**

Date Comparisons

Fields defined in the database as date fields may be directly compared to each other. To compare a character string to a date field, either the date field has to be converted to a string or the string has to be converted to a date field. For example, the following statements are identical in function:
REV_DATE = CTOD("01/15/1991")
DTOC(REV_DATE) = "01/1519/91"

Numeric Comparisons

To compare a numeric field against a character string, the numeric field must first be converted to a string using the string function. Make sure that the number of characters in the converted string equals the number of characters in the compare string. See STR function above. For example: *"25.50" = STR(COST,5,2)* returns true if cost contains the value *25.50*.

Sample Query Statements

Condition: **TITLE = "ASSY"**
Purpose: Find all files whose title begins with ASSY. Note that fields defined as strings must be enclosed in quotation marks.

Condition: **"BLDG C" $ TITLE**
Purpose: Find all files whose title contains the text BLDG C.

Condition: **DTOC(REV_DATE) = "03/01/1992"**
Purpose: Find all files whose revision date is March 1, 1992. Note that you have to convert a date field to a character string first before you can compare it to another string.

Condition: **SUBSTR(DTOC(REV_DATE),9,2) = "92"**
Purpose: Find all files whose revision year is 1992. In this example, you first convert the date to a character string, extract the year by using the substring function, and compare it to another string containing the desired year.

Condition: **SUBSTR(DTOC(REV_DATE),1,2) = "12"**
Purpose: Find all files whose revision month is December. Same as above except your are extracting the month portion of the date string.

Condition: **UPPER(DRAWNBY) = "SMITH"**
Purpose: Find all the drawings drawn by draftsperson SMITH. Since the person's name may have been entered in both uppercase and lowercase, it is best to convert the field contents to uppercase letters first before testing.

Condition: **DELETED()**
Purpose: Find all the files which have previously been marked for deletion.

Condition: **PRICE >= 10.00 .AND. PRICE <= 20.00**
Purpose: Find all the products with prices between ten and twenty dollars.

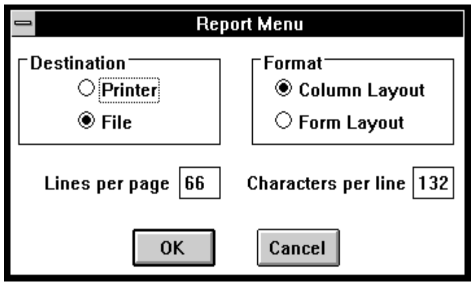
Condition: **REVISION .NOT. OLD**
Purpose: Test if revision is not old where revision is defined as a logical field.

Database Reporting

Database ? Report command

Use this command to report on the contents of your database.

Upon entry, Slick! displays the Report Control Screen shown below. You may change the report settings by picking the entry and entering changes in the input box.



Report Menu

Destination: Printer [✓ File]

Format: [✓ Column Layout] Form Layout

Lines per page 66

Characters per line 132

Destination

You may direct your report output to the printer or to a file by selecting the destination options. You may also choose between column or form layout.

Printer

When selected, your report goes directly to the printer. Make sure the printer is in ready mode before printing. Select OK to begin printing.

File

When selected, your report goes to a disk output file. Slick! will prompt for an output file name. Select OK to create the report file.

Format

Column Layout

If selected, the report will list fields horizontally in column fashion. Make sure that the number of characters per line set below is enough to hold all the fields being reported. Hit any key to return to the Report Menu.

Form Layout

If selected, the report will list all the fields vertically. Hit any key to return to the Report Menu.

Lines Per Page

Default is 66. Enter lines per page. Hit any key to return to Report menu.

Characters Per Line

Default is 132. Enter characters per line. Hit any key to return to Report Menu. **Note:** If set to a value greater than print width of your printer, report will be truncated to the right! Hit any key to return to Report menu.

Report Considerations

? If a query condition is loaded or is being processed, only those records which meet the condition will be reported. See section on *Database Query*.

? If a *Layout* has been set or loaded, this affects the way data is reported. See *Set Layout, Load Layout* in the *Database Browse* section.

? If the contents have been sorted on a field, the report will be sorted likewise. See *Database ? Browse ? Utilities ? Sort on...*

Database Import/Export

Overview

This section discusses how you can import and export database files into and from Slick!'s database. These commands are available from the Database menu. If you are using dBase IV, please read the following discussion first.

dBase IV File Compatibility

Slick!s .DBF files may be read directly by dBase IV. If changes are made to Slick!'s database using dBase IV, you must first *reindex* the file before using the database. See *Reindex* command in the *Utilities* section.

? User may define a database structure in dBase IV identical to Slick!'s structure. User need only to rename the database to SLICKDB.DBF, and reindex the file to be useable by Slick!

Import Database

Database ? *Import Database*

Slick! allows the user to import comma delimited files from other database software packages. Make sure that the number of fields in the input file correspond to Slick!'s database structure. Consistency must be maintained with regards to field type, width, and decimal places. Slick! will prompt:

Enter input file name:
Respond with a file name including path and file extension. Select OK when finished. Slick! will append the information to SLICKDB.DBF. Slick! displays:
Import complete; nnn records imported;

? Slick! provides a mechanism for importing attribute data directly from AutoCAD blocks thru the *Import DWG-Link* command. See the section on *Import/Export Drawing Links*.

AutoCAD Attribute Extraction Option

? If you do not wish to use this feature, you may create a comma delimited file (CDF) by using AutoCAD's attribute extraction command *ATTREXTR*. Modify the file to match Slick!'s database structure exactly and then import into Slick!'s database using the *Import Database* command.

Comma Delimited File Example

Let us assume that your structure contains the following fields:

DRAWING NAME, EXTENSION, PATH, ARCHIVE (DRIVE LETTER), NUMBER, TITLE, AND REVISION DATE.

The CDF file you create must contain exactly seven fields separated by a commas. For example,

```
COLUMBIA,DWG,\ACAD\SAMPLE,D,10,Shuttle,199304
15
NOZZLE,DWG,\ACAD\DWGS,C,20,Fire
Nozzle,19930420
```

?? Note that the ARCHIVED field contains the drive letter. In this example columbia.dwg is in drive D while the nozzle.dwg is in drive C.
?? Note that the date is entered in YYYYMMDD format. This is the internal representation of date fields in Slick!
?? Note that the path field does not contain any drive specification.
?? Note that the key fields NAME, EXTENSION, PATH, and ARCHIVED are all in uppercase letters.

TIP: To see the format of an actual CDF file that you can use as a pattern for importing data into your database, use the export command described below to create one! Then view it with a text editor.

Slick! uses a file's path, name, and extension for its index. This is how a database record is associated with a given drawing. If the path information is incorrect, Slick! will not relate the database information to the drawing.

For example: A drawing with the complete name of *D:\MECH\PART1.DWG* should have the following values for the path, name, and extension fields in Slick!:
PATH: \MECH

NAME:	PART1
EXTENSION:	DWG
ARCHIVED:	D

Export Database

Database ? Export Database

You may export information from the Slick! database as a comma delimited file by using this command. You may then pass this file on to other software packages that accept this file format. Slick! will prompt: *Enter output file name:* Respond with an output file name including path and file extension. Select OK when finished.

Slick! will create a file with information contained in SLICKDB.DBF. Slick! displays: *Export complete.*

?? If a display layout is in effect, only fields that are visible will be exported!
?? If a query is in effect, only matching records will be exported!
?? If a sort order is in effect, exported records will be sorted. See *Database ? Browse ? Utilities ? Set sort on...*

Bidirectional Attribute Exchange

This feature allows you to:

· transfer information from the Slick! database to AutoCAD block attributes using the *Export DWG-Link* command from the Database menu

· transfer information from AutoCAD block attributes and/or AutoCAD system variables to the Slick! database using the *Import DWG-Link* command from the Database menu

AutoCAD Block Attributes

AutoCAD block attributes always contain a *tag* and a *value*. You may link an attribute tag with a database field via the optional *DWG-Link* parameter for that field. The drawing link may be specified by modifying the structure of the database. See *Database ? DB Structure*.

Drawing Link Example

Let us assume that your drawing title block has the following attributes assigned to it for the Columbia sample drawing:

Tag Name	**Value**
DWGNO	1000
TITLE	Columbia Space Shuttle
DRAWNBY	John Doe

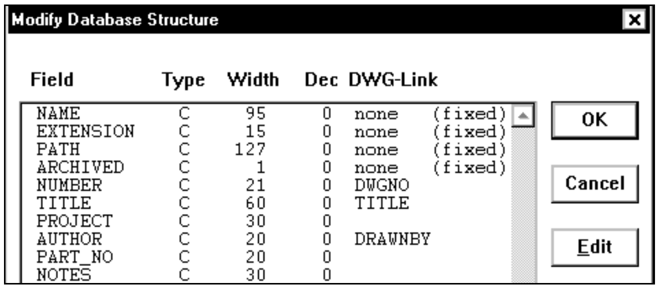
Let us also assume that your database structure contains the following:

Field Name	**Contents**
NUMBER	empty
TITLE	empty
AUTHOR	empty
.....	

Lastly, you want Slick! to copy the values contained in your title block to the corresponding fields in the database without entering the data manually. The first step is to assign a drawing link to the database fields using the *DB Structure* command. You would assign them as follows:

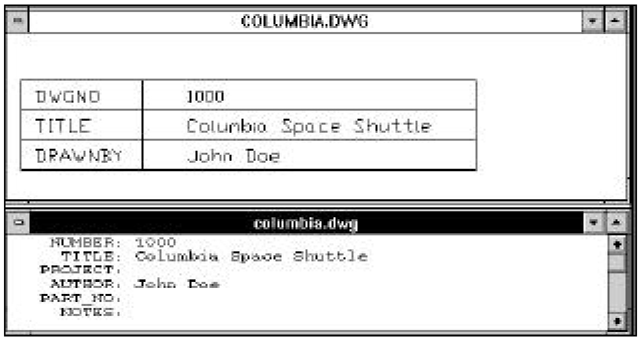
Field Name	Drawing Link
NUMBER	DWGNO
TITLE	TITLE
AUTHOR	DRAWNBY

The modify structure screen should look like the figure below:



Field	Type	Width	Dec	DWG-Link	
NAME	C	95	0	none	(fixed)
EXTENSION	C	15	0	none	(fixed)
PATH	C	127	0	none	(fixed)
ARCHIVED	C	1	0	none	(fixed)
NUMBER	C	21	0	DWGNO	
TITLE	C	60	0	TITLE	
PROJECT	C	30	0		
AUTHOR	C	20	0	DRAWNBY	
PART_NO	C	20	0		
NOTES	C	30	0		

Select the drawing COLUMBIA from the files window. Initially, the database window will be empty since there is no field data. Now, pick *Database ? Import DWG-Link* from the menus. Note now that Slick! will read the attribute values from the drawing and place them in their respective fields. See figure below.

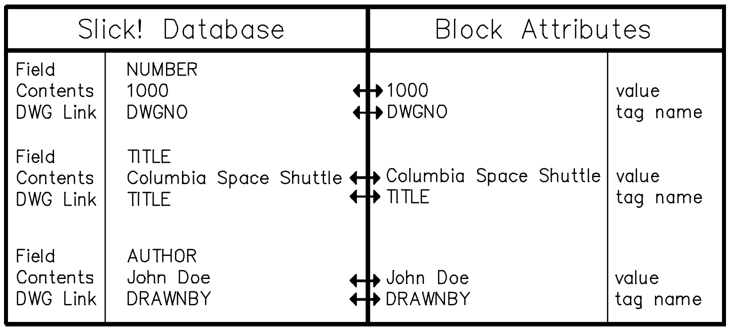


DWGNO	1000
TITLE	Columbia Space Shuttle
DRAWNBY	John Doe

```
  NUMBER: 1000
   TITLE: Columbia Space Shuttle
 PROJECT:
  AUTHOR: John Doe
 PART_NO:
   NOTES:
```

? The database drawing link corresponds to the attribute tag name.
? The database field contents corresponds to the attribute values.

The following chart further clarifies the relationship between attribute blocks and the database:



Slick! Database			Block Attributes	
Field	NUMBER			
Contents	1000	⟷	1000	value
DWG Link	DWGNO	⟷	DWGNO	tag name
Field	TITLE			
Contents	Columbia Space Shuttle	⟷	Columbia Space Shuttle	value
DWG Link	TITLE	⟷	TITLE	tag name
Field	AUTHOR			
Contents	John Doe	⟷	John Doe	value
DWG Link	DRAWNBY	⟷	DRAWNBY	tag name

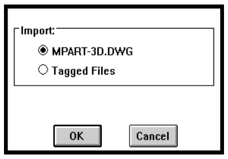
Relationship between Slick! Database and AutoCAD Blocks

Import DWG-Link

Use this command to import attribute information from AutoCAD blocks. See example above. Normally, this command is useful to extract information from a block definition which occurs only *once* in a drawing, such as title blocks.

This command is automatically executed if you have configured Slick! to do so and you have selected a drawing for viewing. See Configure Database.

Slick! reads the AutoCAD drawing file searching for attribute tags with a matching DWG-Link in the database structure. If there are multiple blocks with the same tag name in the drawing, Slick! uses the last attribute value read and places this in the field contents of the database!



Import:
[✓ MPART-3D.DWG]
Tagged Files

Drawing links may be imported for the current file or for previously tagged files. See *Tools ?   Tag* in File Utilities section.

Special drawing links

Slick! provides special drawing links for importing the contents of three AutoCAD system variables:

DWG-Link	ACAD Variable	Description
$$TDCREATE	*TDCREATE*	Date of drawing creation.
$$TDINDWG	*TDINDWG*	Total editing time minutes

$$TDUPDATE	*TDUPDATE*	Date of last update.

If you wish to use any of these drawing links, you must provide field definitions in the database structure to accommodate them. You may specify your own field names but the field type and width must be defined as follows:

Field Name	Type	Width	Decimals	DWG-Link
CREATE_DTE	D	8	0	$$TDCREATE
EDIT_TIME	N	6	0	$$TDINDWG
LAST_UPDTE	D	8	0	$$TDUPDATE

Export DWG-Link

Use this command to export database information to a defined AutoCAD block. By definition, only one database record can exist for a drawing. Therefore, this command is useful to extract information from the drawing database to a block definition which occurs only *once* in a drawing, such as title blocks.

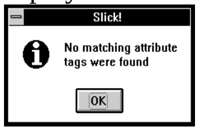
This command creates a script file with the *.SCR* extension in the directory where the the current drawing resides. For example, *D:\ACAD\DWGS\COLUMBIA.SCR*. The script file contains the necessary attribute edit instructions to change the attribute values of linked attribute tags.

To complete the export, you must open the drawing in AutoCAD and run the script command using the script file created by Slick!.

If there are multiple insertions in the drawing of the same tag name, the script file will modify the attribute value of **all occurences** of that tag name encountered!
? **The special drawing links *$$TDCREATE, $$TDINDWG, $$TDUPDATE* may not be exported!**

No matching attribute tags

If Slick! cannot locate matching attribute tags in the drawing, it displays:



Slick!

No matching attribute tags were found