LabVIEW Database Connectivity Toolkit Cheat Sheet

v. 0.9.3

Data Type Mapping

LV DB Toolkit	SQL Data Type	LabVIEW Data Type
string	CHAR(x), VARCHAR(x)	32-bit int (>2147483647), 64-bit int/enum, Boolean, String, Path, I/O Channel
long	INTEGER	8, 16, 32-bit integers or enums (<=2147483647)
single	REAL	Single Numeric
double	DOUBLE	Double Numeric
date/time	DATE, TIME (p)	Date/Time string, Time Stamp
binary	BINARY (n), VARBINARY(n)	Refnum, Complex, Extended, Picture, Array, Cluster, Variant, Digital Waveform, Digital Data, WDT, Fixed-point Numeric

SQL Clauses

	Applicable	Description and Example

FROM `table_defn [options] [table_alias]`	SELECT
	DELETE
		Describes a table to perform an operation on. It might be just a table name or might include full path
		`SELECT FROM modtest`
		`SELECT FROM C:\qcdata\modtest\MT`

WHERE `expr1 comp_op expr2 [logic_op expr3 comp_op expr4]...`	SELECT
	DELETE
	UPDATE
		Specifies conditions that are applied to each row in the table to determine active set of rows.
		`SELECT * FROM mt WHERE (mt.fld1 = 3 AND mt.fld2 <366)`

GROUP BY `col_expr{, col_expr,...}`	SELECT	Specified one or more column expression to use to group active set rows `SELECT * FROM mt WHERE (mt.fld1 = 3) GROUP BY mt.fld4`

HAVING `expr1 comp_op expr2`	SELECT
	+
	GROUP BY
		Specifies conditions to apply to active set row groups. `GROUB BY` must be specified first!
		`SELECT * FROM mt GROUP BY mt.f4 HAVING AVG (mt.f5) > 234`

ORDER BY `{sort_expr [DESC or ASC]}...`	SELECT	Use to specify row order in the active set of rows and/or groups. Use DESC or ASC for order. `SELECT * FROM mt WHERE (mt.f1 = 3) ORDER BY mt.f2 DESC`
FOR UPDATE OF `col_name1[, col_name2, ...]`	SELECT	Use to lock columns in selected rows for updates or deletion `SELECT * FROM mt WHERE (mt.f2 < 366) FOR UPDATE OF mt.f1, mt.f3`

`SELECT *` **FROM** `C:\gcdata\modtest\MT` **WHERE** `(MT.f1 = 3 AND MT.f2 <=365)` **GROUP BY** `MT.f4` **HAVING** `AVG (MT.f5) >=2344.56` **ORDER BY** `MT.f2 DESC`

SQL Functions

	Date/Time
TIME()	returns current time of day as character string
DATE, TODAY()	returns current date in date format
DATEVAL(char_expr)	converts char_expr to date format
MONTH, **DAY**, **YEAR**(date_expr)	returns month /day / year part of date_expr as a number
CTOD(char_expr, fmt)	converts char_expr to date format using fmt template*
DTOS(date_expr)	convert from date_expr to character string using format YYYYMMDD
DTOC(date_expr, fmt_value[,'separator_char'])	convert date_expr from date to chars string using fmt templ** and (opt) separator (/) *templates: 0 => MM/DD/YY, 1 => DD/MM/YY, 2 => YY/MM/DD, **10 => MM/DD/YYYY, 11 => DD/MM/YYYY, 12 => YYYY/MM/DD
	String
CHR(number_expr)	character having ASCII value number_expr
LOWER(char_expr)	force all to lower case in char_expr
LTRIM, **RTRIM**, **TRIM**(char_expr)	remove leading /trailing / both blanks from char_expr
SUBSTR(char_expr, number_expr1,number_expr2)	substring of char starting at character number number_expr1 of length number_expr2
UPPER (char_expr)	force all letters in char_expr to upper case
LEFT, **RIGHT**(char_expr)	leftmost / rightmost character in char_expr
SPACE(number_expr)	generate a string of number_expr blanks
STR(number_expr,width[,precision])	converts number_expr to a character string of width and optional precision digits
STRVAL(expr)	converts any expr to a character string
LEN(char_expr)	number of characters in char_expr
NUMVAL(char_expr)	converts char_expr to a number (if valid expr)
VAL(char_expr)	converts char_expr to a number
	Math
ROUND(number_expr1,number_expr2)	number_expr1 rounded to number_expr2 decimal places
POWER(number_expr1, number_expr2)	raises number_expr1 to number_expr2 power
INT(number_expr)	returns integer part of number_expr
MOD(number_expr1,number_expr2)	divides number_expr1 by number_expr2 and returns remainder
ABS(x)	
	Aggregate (Use with GROUP BY)
MAX, **MIN**(column_expr)	maximum / minimum value of column_expr
MAX, **MIN**(number_expr1, number_expr2)	larger /smaller of number_expr1 and number_expr2
AVG(numeric_column_name)	average of all non-NULL values in numeric_column_name
COUNT(*), **COUNT**(expr)	number of all rows in a table
SUM(column_expr)	sum of values in column_expr

Full list here: http://oreilly.com/catalog/sqlnut/chapter/ch04.html

SQL Operators

Constants	`123,`
	`'abcd', "abcd",`
	`{9/27/71}, {14:32:56},`
	`.T., .F.`

Numeric	`( ) + - * / + - ** ^`
	`(A + B) * (C - D) / F * (A**B)-A^B`

Character	`+ 'txt a ' + 'txt b' -> 'txt a txt b'`
	`- 'txt a ' - 'txt b' -> 'txt atxt b'`

Comparison	**`= <> >= <= IN, NOT IN, ANY, ALL,`**
	`BETWEEN, EXISTS, [NOT] LIKE, [NOT] NULL`
	`WHERE a IN ('apple','orange')`
	`WHERE EXISTS (SELECT ...)`
	`WHERE a LIKE 'tar%'`

Date	`+ -`
	`testdate - {1/30/18} -> number of days`

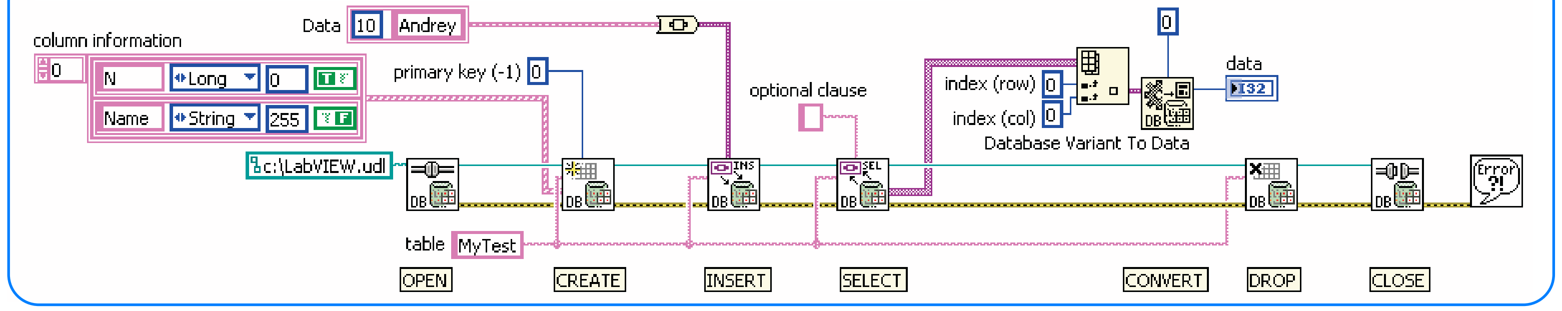
Logical	**`(), NOT, AND, OR`**
	`WHERE (a AND b) OR (c AND d)`
	`WHERE NOT (a IN (SELECT...))`

Set	**`UNION`** (set all rows from all individual distinct queries) `SELECT ... UNION SELECT ...`
Other	**`*`** (all columns) `Select * FROM Table1`

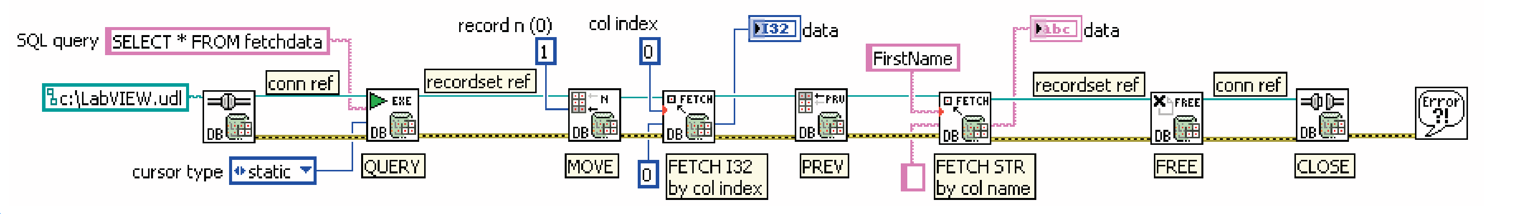
UDL File Format

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Teste.mdb;Mode=ReadWrite;Persist Security Info=False
```

Create / Write / Retrieve Data to / from Database (basic usage)



Fetch Data + navigate (advanced usage)



ADO Reference Classes

Connection
Use this class to define the database connection parameters, such as the OLE DB provider used, the connection string used, and the default database used.

Command (C)
Use this class to execute commands and capture parameters returned from query or stored procedures. Create a Command reference by first creating a Connection reference and then calling the DB Tools Create Parameterized Query VI. You can get or set properties related to the command or the parameters associated with the command.

Recordset (R)
Use this class to manipulate data. Create a Recordset reference by creating a Connection reference and then calling DB Tools Execute Query VI. You can get or set properties related to the column information, the number of records available, the beginning or end of file markers, and the type of cursor used..

Command-Recordset (CR)
Use this class for situations where commands and recordsets are used together, such as SQL queries. Create a Command-Recordset reference by first creating Connection and Command references and then calling DB Tools Execute Query VI. You can get or set all the properties available to the Command and Recordset references.

Warning: After you create a reference, delete it with the DB Tools Free Object!

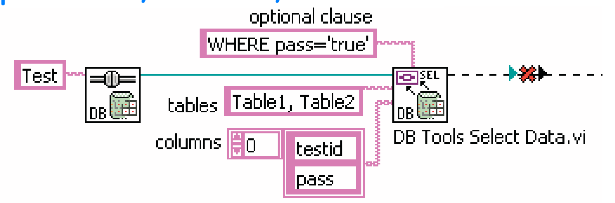
Cursor Types

Forward-only — default and permits only forward movement through the recordset.Any changes made to the database by other users during navigation will not be seen. Forward-only cursors are dynamic because detection of changes occurs as the current row is
processed. This is a high-performance cursor that uses the least resources.
Keyset — allows forward and backward navigation.You can see records added by other users, but records deleted by others will not be removed from view.
Dynamic — allows forward and backward navigation. You can see all changes made to the database, locally and by other users. Use the dynamic cursor if your application must detect all concurrent updates made by other users.
Static — allows forward and backward navigation with no ability to see any changes made by other users during navigation. The static cursor always displays the result set as it was when the cursor was first opened. Use the static cursor if your application does not need to detect data changes and requires scrolling.

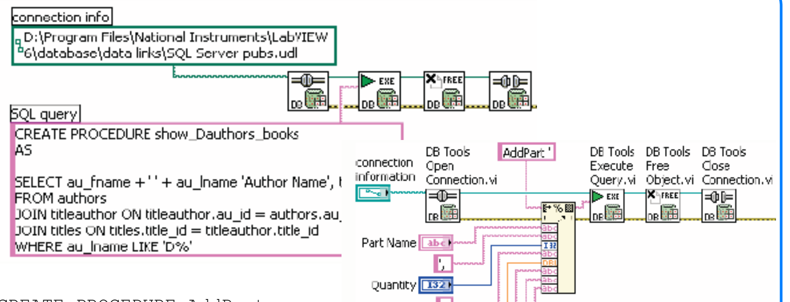
Glossary

ActiveX Microsoft's Object Linking and Embedding (OLE) that allows components to be embedded
ADO (ActiveX Data Objects) Microsoft API that is designed as the Microsoft standard for data access.(COM object)
ADTG (Advanced Data TableGram) ProprietaryMicrosoft binary file format. (Compact and faster than XML)
API (Application Programming Interface) Communicate with OS or control program such as DBMS
BOF EOF (Beginning/End Of File) Marker that points just before first/after last record in a database table
BCD (Binary Coded Decimal) - Each digit converted to binary. 12-digit number would take 12 bytes
BLOB (Binary Large OBject) - Used by Oracle. Array, Waveforms and clusters stored as BLOB (images)
DAO (Data Acess Objects) API for data access using M$ Access. Can be DAO/Jet or DAO/ODBC.
DCL (Data Control Language) Component of SQL that protect databases from harm such as lock
DDL (Data Definition Language) Component of SQL that cretaes, modifies or deletes databases
DML (Data Manipulation Language) Component of SQL that operated on data within the database
DSN (Data Source Name) - Way to refere specific database (use ODBC Admin in Windows for create DSN)
  System DSN - stored in Windows Registry. available to all users. (HKLM/Software/ODBC/ODBC.INI)
  User DSN - available to particular user (HKCU/Software/ODBC/ODBC.INI)
DBMS (Database Management System) For example, Oracle or SQL Server are DBMS
ISAM (Indexed Sequential Access Method) - DB with index. (M$ Jet DB) Paradox, dBase, Btrieve, Excel FoxPro.
MDB (Microsoft Access Databases)
COM (Component Object Model) - Component software architecture from M$. Provides the interfaces between obj.
DCOM (Distibuted Component Object Model) - same as COM but allows run remotely
MDAC (Microsoft Data Access Component) - M$ UDA Strategy. includes ODBC, OLE DB and ADO. (v 2.5)
Jet database engine - underlying DBMS of MDB and numerous ISAM
ODBC (Open Database Connectivity) - M$ API. Use Win Control Panel ODBC utility to specify db connections.
OLE DB (OLE DataBase) - allows low-level access to ODBC databases
  OLE DB Consumers/Data/Service Providers - Three general types of COM components for OLE DB
OCI (Oracle Call Interface) - Oracle's native API for connect to Oracle DBMS
RDBMS (Relational Database Management System) - DBMS that uses relational DB.
SQL (Structured Query Language) - used to process data in relational DB. Originally dev by IBM.
UDA (Universal Data Access) Technology from M$. OLE DB is system interface of UDA and ADO is API.
UDL (Universal Data Link) - File defines data source. UDL contains info about what OLE DB provider is used.
XML (eXtensible Markup Language) - supported by the Database Connectivity Toolkit.

Multiple Tables, columns, conditions



Stored Procedure



connection info
D:\Program Files\National Instruments\LabVIEW 6\database\data links\SQL Server pubs.udl
SQL query
connection information
DB Tools Open Connection.vi
AddPart '
DB Tools Execute Query.vi
DB Tools Free Object.vi
DB Tools Close Connection.vi
Part Name
Quantity
Price
Description

```
CREATE PROCEDURE show_Dauthors_books
AS

SELECT au_fname + ' ' + au_lname 'Author Name', t
FROM authors
JOIN titleauthor ON titleauthor.au_id = authors.au_
JOIN titles ON titles.title_id = titleauthor.title_id
WHERE au_lname LIKE 'D%'
```

```
CREATE PROCEDURE AddPart
@part_name char(40),
@part_qty int,
@part_price money,
@part_descr varchar(255) = NULL
AS
INSERT parts (name, qty, price, description)
VALUES (@part_name, @part_qty, @part_price, @part_descr)
```

Database:

DB Tools Open Connection.vi

DB Tools Close Connection.vi
Don't forget it!

DB Tools Create Table.vi
Creates a new table in the database

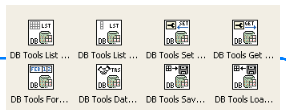
DB Tools Drop Table.vi
Deletes the specified **table**

DB Tools Insert Data.vi
Inserts a new row into the table

DB Tools Select Data.vi
Selects data from the **table**

Database Variant To Data
Converts a **database variant** to the LabVIEW data type

Utility:



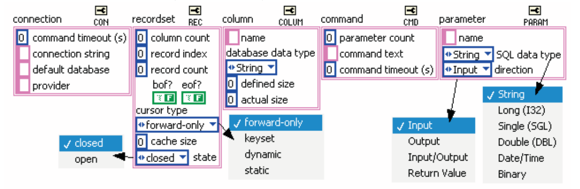
DB Tools List Tables.vi

DB Tools List Columns.vi
Lists the columns present in **table**.

Get/Set Properties

DB Tools Get Properties.vi
Gets properties of the object
(Recordset(R), Recordset (CR), Column (R), Column (CR),
Command (C), Command (CR), Parameter (C), Parameter (CR),

DB Tools Set Properties.vi
Sets properties on the object
(Connection, Command (CR), Parameter (CR), Command (C), Parameter (C)
references: R-recordset, C-command,CR-command-recordset



Load/Save

DB Tools Save Recordset To File.vi

DB Tools Load Recordset From File.vi

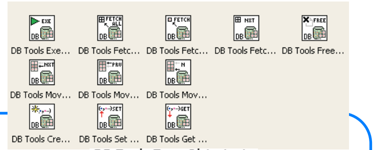
Saves the recordset identified by the **recordset or command-recordset reference** to either an XML or ADTG file. The ADTG file format is a proprietary format that only the LabVIEW can interpret. The ADTG format results in a smaller file than the XML format. (C or CR)

Misc functions

DB Tools Format Datetime Str.vi
Returns a string containing the formatted date and time from seconds (now)

DB Tools Database Transaction.vi
Begins, commits, or rolls back a transaction for any type of reference (connection, comm(C), comm(CR))

Advanced:



Queries

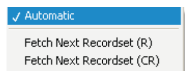
DB Tools Execute Query.vi
Executes an SQL query and returns a recordset reference
ADO Connection or Command instances!

DB Tools Free Object.vi
Frees an object by destroying its associated reference

DB Tools Create Parameterized Query.vi
Creates a parameterized **SQL query** and returns a **command reference**

Fetch

DB Tools Fetch Recordset Data.vi
Retrieves the data in the recordset identified by the **recordset or command-recordset reference**



DB Tools Fetch Next Recordset.vi
Retrieves the next recordset in a multi-recordset query identified by **recordset or command-recordset reference**.

DB Tools Fetch Element Data.vi
Retrieves the data located at the **column index** of the current record in the recordset identified by the **recordset reference**. The **column index** can be either the zero-indexed position of the column in the recordset or the name of the column.
String, Integer, Single, Double, Date/Time Binary (all of them R or CR)

Move

DB Tools Move To Next / Previous / Record N .vi
Moves the cursor in the recordset.
To move the cursor to the last record, set **n** to –1
Be careful with forward-only cursor!

Params

DB Tools Get Parameter Value.vi

DB Tools Set Parameter Value.vi

Retrieves / Sets the value of the parameter specified by the **parameter index** input in the command or command-recordset. The **parameter index** can be either the zero-indexed position of the parameter in the command or the name of the parameter.

©2009. Andrey Dmitriev. Some rights reserved. This page is licensed under a Creative Commons Attribution License. NI and LabVIEW are trademarks of National Instruments Corporation.