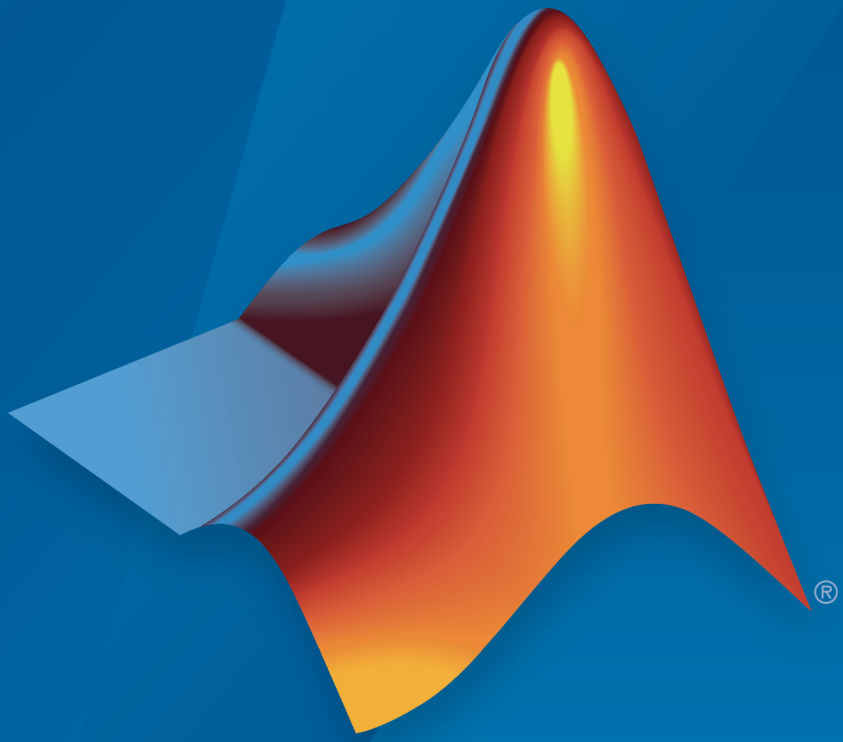
Database Toolbox™ Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Database Toolbox™ Release Notes

Contents

R2016b

R2016a

R2015b

R2015a

Bug Fixes

R2014b

R2014a

Bug Fixes

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

R2006b

R2006a

Bug Fixes

R14SP3

R14SP2

Bug Fixes

R2016b

Graph Database Interface: Retrieve graph data from Neo4j

To import graph data in a Neo4j® database into MATLAB®, use the MATLAB interface to Neo4j. To perform graph network analysis with graph data in MATLAB using the `digraph` object, create a Neo4j database connection. Or, you can explore the graph using MATLAB functionality. If you are familiar with the Cypher® query language, you can execute Cypher queries. For details about using the MATLAB interface to Neo4j, see "Graph Database".

`DatabaseDatastore` functionality and object properties changes

To analyze data using common MATLAB functions, such as `mean` and `histogram`, you can create a tall array using the `DatabaseDatastore` object. For details about using tall arrays with Database Toolbox™, see "Analyze Large Data in Database Using Tall Arrays".

To create a `DatabaseDatastore` object, use `databaseDatastore` instead of `datastore`.

The `DatabaseDatastore` object has two additional object properties. The `VariableNames` property provides the variable names of the retrieved data table. The `ReadSize` property specifies the number of rows to read from the retrieved data table.

The `read`, `readall`, and `preview` functions return data as a `table`.

Compatibility Considerations

- The `DatabaseDatastore` object properties have changed:
  - The `Cursor` property has been removed.
  - Two properties are added: `VariableNames` and `ReadSize`.

  For details about the object properties, see `DatabaseDatastore`.
- This syntax for the `read` function has been removed.

  ```
  data = read(dbds,rowcount)
  ```

  To specify the number of rows to retrieve, set the `ReadSize` property of the `DatabaseDatastore` object instead.

- The data retrieval functionality has changed:
  - The functions retrieve data as a `table` by default. Setting the database preference `DataReturnFormat` to `'table'` is not required.
  - `read` and `preview` throw an error when there is no more data in the `DatabaseDatastore` object for reading.
  - If `read` finds no more data to read, `hasdata` returns logical 0.

Functionality being removed or changed

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`bestrowid`	Errors	Nothing	No replacement
`clearwarnings`	Errors	Nothing	No replacement
`crossreference`	Errors	Nothing	No replacement
`driver`	Errors	Nothing	No replacement
`drivermanager`	Errors	Nothing	No replacement
`isdriver`	Errors	Nothing	No replacement
`isjdbc`	Errors	Nothing	No replacement
`isnullcolumn`	Errors	Nothing	No replacement
`isurl`	Errors	Nothing	No replacement
`register`	Errors	Nothing	No replacement
`unregister`	Errors	Nothing	No replacement
`versioncolumns`	Errors	Nothing	No replacement
`rsmd`	Still runs	Nothing	No replacement
`resultset`	Still runs	Nothing	No replacement
`tables(conn)`	Errors	Use syntaxes with at least two input arguments instead.	Remove all instances of the `tables(conn)` syntax. Replace these instances with any of the remaining syntaxes. For details, see `tables`.

R2016a

MATLAB Interface to SQLite: Create, read, and write data from SQLite database files without external drivers and administration

To import and update data without an existing database or database administration, use the MATLAB Interface to SQLite. For details, see Working with the MATLAB Interface to SQLite. For the supported functions, see this table.

Function	Purpose
sqlite	Create SQLite connection.
exec	Perform database operation in the SQLite database file.
fetch	Run SQL query and import data from the SQLite database file.
insert	Export data into the SQLite database file.
close	Close SQLite connection.

`fetch` Function Speed Improvement: Import data faster using the JDBC driver

When you connect to a database using the JDBC driver, importing data is faster using the fetch function.

Support for 32-bit Windows removed

The Database Toolbox no longer supports connection to a database using a 32-bit driver.

Compatibility Considerations

Use a 64-bit database. Or, install a 64-bit driver that works with the 32-bit database. For details, consult with your database administrator.

For Microsoft® Access™, see http://www.mathworks.com/matlabcentral/answers/235949-how-to-connect-to-32-bit-microsoft-access-database-from-64-bit-matlab.

Functionality being removed or changed

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`clearwarnings`	Warns	Nothing	No replacement
`versioncolumns`	Warns	Nothing	No replacement
`crossreference`	Warns	Nothing	No replacement
`bestrowid`	Warns	Nothing	No replacement
`isnullcolumn`	Warns	Nothing	No replacement

R2015b

ODBC Interface Functions: Export and retrieve database information using native ODBC connections

More Database Toolbox functions support the native ODBC interface for exporting data and retrieving database information and metadata. For a list of supported functions, see Connecting to a Database Using the Native ODBC Interface.

Read and Write Performance Improvements: Import and export data more quickly

Data import and export functions can retrieve and write data faster. Particularly, datainsert and update call the `TRANSACTION` command of SQL to insert or update records faster for these databases: Microsoft SQL Server®, MySQL®, Oracle®, and PostgreSQL.

Data Export Functions: Insert or replace data using table, structure, and dataset arrays

The datainsert and update functions can export data in tabular, `dataset`, and structure arrays to databases.

Functionality being removed or changed

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`driver`	Warns	Nothing	No replacement
`drivermanager`	Warns	Nothing	No replacement
`isconnection`	Warns	`isopen`	Replace all instances of `isconnection` with isopen.

R2015a

Version: 5.2.1

Bug Fixes

R2014b

Version: 5.2

New Features

`DatabaseDatastore` for applying `mapreduce` to data contained in relational databases

Create a DatabaseDatastore to work with large amounts of data in relational databases. Write custom functions to implement `mapreduce` to process large amounts of data. To create a `DatabaseDatastore`, you must create a `DatabaseDatastore` object. This object is a type of datastore.

Function	Purpose
datastore	Create a `DatabaseDatastore`.
hasdata	Determine if a `DatabaseDatastore` contains more data in the cursor object.
preview	Display the first eight records in a `DatabaseDatastore`.
read	Read data in a `DatabaseDatastore`.
readall	Read every record in a `DatabaseDatastore`.
reset	Reset the cursor position in a `DatabaseDatastore`.

Scrollable cursors for accessing data using relative and absolute position inputs

Fetch data sequentially or scroll up or down in the data without executing the query again. Scrolling within the data offers advantages when you are working with a large data set. An advantage of scrollable cursors is reading data in the middle of a large data set using the cursor position offset.

Create a scrollable cursor using exec. Retrieve data from a scrollable cursor using fetch. Use relative and absolute position inputs in `fetch` to retrieve data starting from a specific location in the data set.

R2014a

Version: 5.1

Bug Fixes

R2013b

Version: 5.0

New Features

Bug Fixes

Fast access to ODBC connections via a native ODBC driver

Support for native ODBC database connection for Windows® platforms. The native ODBC interface is available only for the command line. To use this interface, see Using the Native ODBC Database Connection. The native ODBC interface supports the following functions:

- database
- fetch
- exec
- insert
- fastinsert
- close

`table` data type support

You can return a `table` data type rather than a cell array. Use the setdbprefs command to set the database preference for the `DataReturnFormat` property to `'table'`.

R2013a

`fetch` function accepts user-defined batch sizes

setdbprefs is updated with new properties (`FetchInBatches` and `FetchBatchSize`) that support fetch when requesting large data.

R2012b

Database Explorer app for interactively exchanging data with databases

dexplore starts Database Explorer, which is the Database Toolbox GUI for connecting to a database and importing data to the MATLAB workspace. Alternatively, you can start Database Explorer by selecting **Database Explorer** from the **Database Connectivity and Reporting** section of the **Apps** tab in the MATLAB Toolstrip.

Functionality Being Removed or Changed

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`querybuilder`	Warns	`dexplore`	Continue to use `querybuilder` for exporting data.

R2012a

Version: 3.11

New Features

Bug Fixes

Execute .SQL Files

The new runsqlscript function lets you execute SQL commands from a `.SQL` file on a connected database, and store the results in a `cursor` array. You can input the results from executing `runsqlscript` to functions that accept `cursor` array inputs.

Improvements to the Database Constructor

When using a JDBC driver, you can input individual connection properties to the database constructor, database.

R2011b

Preferences Now Persistent Across MATLAB Sessions

The preferences you set using the Preference dialog box or the setdbprefs function now persist across MATLAB sessions.

Compatibility Considerations

In releases before R2011b, if you changed your preferences during a MATLAB session, these preferences would not remain in the next MATLAB session.

Change in Behavior for the update Function

update lets you update images, Booleans, doubles, and strings in a manner consistent with fastinsert.

Warning and Error ID Changes

Many warning and error IDs have changed from their previous versions. These warnings or errors typically appear during a function call.

Compatibility Considerations

If using warning or error IDs, you might need to change the strings you use. For example, if you turned off a warning for a certain ID, the warning might now appear under a different ID. If you use a `try/catch` statement in your code, replace the old identifier with the new identifier. There is no definitive list of the differences, or of the IDs that changed.

R2011a

Version: 3.9

New Features

Bug Fixes

New datainsert Function Exports MATLAB Cell Array Data into a Database Table

The new datainsert function inserts data from the MATLAB workspace into a database table, much like the fastinsert function. The new `datainsert` function is faster.

R2010b

Version: 3.8

New Features

Bug Fixes

Now Possible to Import Data into MATLAB Dataset Object

If you have Statistics Toolbox™ installed, you can now return a `dataset` object rather than a cell array. Use the `setdbprefs` command to set the database preference for the `DataReturnFormat` property to `'dataset'`.

R2010a

Version: 3.7

New Features

Bug Fixes

New Connection Object Methods

Several new connection object methods provide database-specific information. The new methods are:

- database.catalogs
- database.columns
- database.schemas
- database.tables

See the individual reference pages for more information on how to use these methods.

Enhanced Error Messages

New enhanced error messages provide more information about the error. For example, the 2009b error message `Drivers not Found/Loaded` is now `Drivers not Found/Loaded. Please verify that login information and database url are valid` in 2010b. This error will appear when the driver input is valid but the database URL is invalid.

Improved Write Performance

New bulk insert code templates provide significant performance upgrades.

R2009b

Version: 3.6

Bug Fixes

R2009a

Version: 3.5.1

Bug Fixes

R2008b

Version: 3.5

Bug Fixes

R2008a

Version: 3.4.1

Bug Fixes

R2007b

Version: 3.4

Bug Fixes

R2007a

Version: 3.3

New Features

Bug Fixes

setdbprefs Accepts Structure Input

The `setdbprefs` function now accepts a structure as input. For example, you can run the following commands to assign values to `s`:

```
s.DataReturnFormat = 'numeric';
s.ErrorHandling = 'report';
```

You can also do this for other `setdbprefs` properties whose values you want to change. Then set the preferences using the values in `s` by running the command:

```
setdbprefs(s)
```

For more information, see the setdbprefs reference page.

Visual Query Builder Generated M-File Includes Placeholder for Password and Assigns Preferences to Structure

When you run a query in the Visual Query Builder and select **File > Generate M-File**, the resulting M-file now includes a placeholder string `password` in the `database` statement. If a password is required for the connection, such as for connections established via JDBC drivers, substitute the password for the `password` string. If no password is required, the M-file will run as is. For more information, see About Generated Files.

The generated M-file assigns values for the preferences to the structure `s`. For more information, see the setdbprefs reference page.

Preference Added for Temporary Registry Output; Ensures Full Output for getdatasources

When you use getdatasources to view the data sources for your system, ensure that you view all data sources by specifying a temporary, writable, output directory using the new preference, `TempDirForRegistryOutput`. This is useful when you add data sources and do not have write access for the MATLAB current directory, where the toolbox temporarily writes ODBC registry settings. Without write access, `getdatasources` does not always return data sources you added. In that event, run setdbprefs to specify a value for the `TempDirForRegistryOutput` preference, where the value is the full path name to a directory for which you have write access.

R2006b

Enhanced fetch Combines exec with Existing fetch

The new function, `database.fetch`, executes the specified SQL query and imports results into the MATLAB workspace, given the connection handle `conn`. It is provided for convenience, to combine capabilities of the existing `exec` and `cursor.fetch` functions. In statements and code, do not specify `database.fetch` or `cursor.fetch` but rather, just specify `fetch` with the appropriate objects provided as arguments; the toolbox runs `database.fetch` or `cursor.fetch` as appropriate based on the arguments.

Unlike `cursor.fetch`, `database.fetch` does not return a cursor object on which you can run subsequent Database Toolbox functions, but rather returns all data to a MATLAB variable. For more information about `database.fetch` and how it differs from `cursor.fetch`, see the fetch reference page, as well as the database.fetch and cursor.fetch reference pages.

Import Data from Multiple Resultsets

The new function, fetchmulti, imports data into the MATLAB workspace from multiple resultsets, which you retrieve via an `exec` call to a stored procedure that contains two or more `select` statements.

Run Stored Procedures to Return Output Parameters

The new function, runstoredprocedure, executes a stored procedure using input parameters specified in a cell array to return output parameters. This allows you to retrieve the value of a variable into a MATLAB variable. `runstoredprocedure` overcomes a limitation of `exec`; when you run a stored procedures via `exec`, you can only retrieve resultsets.

Specify Catalog and Schema Using Visual Query Builder

You can now specify the catalog and schema for a data source using the Visual Query Builder. The `default` is none, meaning you do not need to select values for them.

Preferences Option to Find Additional Data Sources

The new `setdbprefs` option, `UseRegistryForSources`, instructs the Visual Query Builder to search the Microsoft Windows registry to find any ODBC data sources not uncovered using the system `ODBC.INI`.

MATLAB Change to Assignment of Nonscalar Structure Array Fields Might Impact Database Toolbox Users

In Version 7.3 (R2006b) of the MATLAB software, a change was made to how a nonscalar structure array field is assigned to a single MATLAB variable. For more information, see Assigning Nonscalar Structure Array Fields to a Single Variable in the MATLAB Release Notes.

Compatibility Considerations

As a result of this change in the MATLAB software, you may need to modify your Database Toolbox M-files.

R2006a

Version: 3.1.1

Bug Fixes

R14SP3

Version: 3.1

New Features

Bug Fixes

fastinsert Function Added

There is a new function, fastinsert, that you can use instead of the insert function to export data about three times more quickly than `insert`. It also allows exporting for all object types, so that any data you can retrieve from a database you now can export to the database, including binary objects.

While there are no known problems with `fastinsert`, if you receive unexpected results, return to using `insert` and report the problem with `fastinsert` via Technical Support.

Note that the Visual Query Builder insert feature uses the `insert` function instead of `fastinsert`.

JDBC Drivers Now Supported for Visual Query Builder on Microsoft Windows Systems

You now can use the Visual Query Builder (VQB) with JDBC drivers on Windows platforms. Previously, only ODBC drivers were supported.

The confds function now displays an enhanced dialog box you use to define JDBC data sources. With it, you save and load data source information via MATLAB MAT-files.

For details, see Setting Up JDBC Data Sources in the Database Toolbox documentation.

Define Data Sources from Within the Visual Query Builder

The Visual Query Builder now includes two new items under the **Query** menu:

- **Define ODBC Data Source**—Directly access your Windows ODBC Data Source Administrator dialog box where you define ODBC data sources.
- **Define JDBC Data Source**—Access the Define JDBC Data Source dialog box for defining JDBC data sources to use with the VQB. The function equivalent is `confds`. When you define a JDBC data source, the information is saved in a MAT-file so you can use it again in a later session. Later, open the MAT-file using the Define JDBC Data Source dialog box, or using `setdbprefs('JDBCDataSourceFile','fullpathtomatfile')`.

For details, see Configuring Your Environment in the Database Toolbox documentation.

setdbprefs Function Enhanced

New arguments are supported for defining the JDBC data source MAT-file. For details, see the setdbprefs reference page.

Dynamically Add JDBC Drivers File

You can dynamically add a JDBC drivers file to the MATLAB Java® `classpath` using the MATLAB `javaaddpath` function. You can use this method instead of adding a pointer to the JDBC drivers file in your `classpath.txt` file. The advantage of using `javaaddpath` is that you do not have to restart the MATLAB software session after running the `javaaddpath` statement. The disadvantage is that this only applies to the current session and so you need to run the `javaaddpath` statement in each new session. For details, see Setting Up JDBC Data Sources in the Database Toolbox documentation.

64-Bit FLOAT for Microsoft SQL Server Software Is Fully Supported

You now can retrieve 64-bit `FLOAT` data using Microsoft SQL Server software.

Generate M-File from VQB

After running a query using the Visual Query Builder, you can generate an M-file consisting of Database Toolbox functions that perform the query. This is useful if you know how to run queries with the VQB and want to determine the equivalent functions, particularly the SQL statements in `exec` and `insert`.

update Function Enhanced to Export Multiple Records

The `update` function has been enhanced so that you can export multiple records based on different `where` clauses. The number of `where` clauses must equal the number of records in the array of data you are exporting. For details, see the reference page for update.

logintimeout Function Now Supported on Linux Platforms

The `logintimeout` function is now supported on Linux® platforms.

R14SP2

Version: 3.0.2

Bug Fixes