1

DBD::SearchServer

Version

Version 0.20.

This driver was previously known as `DBD::Fulcrum`.

Author and Contact Details

The driver author is Davide Migliavacca. He can be contacted via the *dbi-users* mailing list. Davide Migliavacca has no relationship with PCDOCS/Fulcrum, the maker of SearchServer, and particularly no contact with product support for PCDOCS/Fulcrum customers.

Supported Database Versions and Options

The `DBD::SearchServer` module supports PCDOCS/Fulcrum SearchServer, versions 2.*x* thru 3.5.

Fulcrum SearchServer is a very powerful text-retrieval system with a SQL interface. You should not expect to find a full-fledged SQL RDBMS here. Refer to the product documentation for details about the query language.

Connect Syntax

Under UNIX, you may specify where SearchServer will find the database tables by using a set of environment variables: FULSEARCH, FULCREATE, and FULTEMP. So the connect string is always just:

```
dbi:SearchServer:
```

Under WIN32, you may use the fully qualified DSN syntax using the ODBC data source name as the third component of the connect string:

```
dbi:SearchServer:DSN
```

There are no driver specific attributes for the `DBI->connect()` method.

Numeric Data Handling

SearchServer has two numeric data types: INTEGER and SMALLINT. INTEGER (or INT) is an unsigned 32-bit binary integer with 10 digits of precision. SMALLINT is a signed 16-bit binary integer with 5 digits of precision.

String Data Handling

SearchServer supports the following string data types:

```
CHAR(size)
VARCHAR(size)
APVARCHAR(size)
```

A CHAR column is of fixed size, whereas a VARCHAR column can be of varying length up to the specified maximum size. If the size is not specified, it defaults to 1. The maximum size for a CHAR or VARCHAR column is 32,767.

APVARCHAR is a special data type. You can have at most one APVARCHAR column per table; it is designed to contain the full text of the document to be indexed and it is used in queries to retrieve the text. It is eventually modified to identify spots where the query matched. The maximum length of the APVARCHAR column is 2,147,483,647.

The CHAR type is fixed length and blank padded to the right.

SearchServer has its own conversion functionality for national language character sets. Basically it treats all text as being specified in one of three internal character sets (FTICS). It is up to the application to use character sets consistently. The document readers (software that is used by SearchServer to actually access documents when indexing) are responsible for translating from other characters sets to FTICS. A number of ''translation'' filters are distributed with the product.

ISO Latin 1 (8859-1) is supported. See the Character Sets section of the SearchSQL Reference manual for more details on character set issues.

Date Data Handling

SearchServer supports only a DATE data type. A DATE can have any value from January 1, 100 AD to December 31, 2047 AD with one day resolution. Rows in tables have an automatic read-only FT_TIMESTAMP column with a better resolution, but it is not of a DATE type (it is an INTEGER). Also, only date literals can be used with DATE columns.

The date format is `YYYY-MM-DD` (ISO standard). There are provisions for other formats but their use is discouraged.

Only the ISO date format is recognized for input.

If a two digit year value is entered, then 1900 is added to the value. However, this isn't supported functionality, for good reason.

No date time arithmetic or functions are provided and there is no support for time zones.

LONG/BLOB Data Handling

The APVARCHAR type can hold up to 2 gigabytes.

LongReadLen and *LongTruncOk* are ignored due to very different semantics of the APVARCHAR type.

You need to use the undocumented `blob_read()` method to fetch data from an APVARCHAR column. Inserting an APVARCHAR column happens indirectly by specifying an external document in the FT_SFNAME reserved column. Document data is not really inserted into the tables, it is indexed. Later, however, you can fetch the document selecting the APVARCHAR column.

Other Data Handling issues

The `DBD::SearchServer` driver does not support the `type_info()` method.

Transactions, Isolation and Locking

`DBD::SearchServer` does not support transactions.

Locking is performed based on the characteristics of the table, set at creation time or modified later with an external utility, `ftlock`.

By default, ROWLOCKING is applied, which applies ''transient'' locks during normal operations including select, searched update, and delete. These locks should not prevent reading the affected rows, but will block additional concurrent modifications, and prevent reindexing of the locked rows.

If set to NOLOCKING, no locking will be performed on that table by the engine, meaning data integrity is left for the application to manage. Please read the docs carefully before playing with these parameters; there is additional feedback with the PERIODIC or IMMEDIATE indexing mode.

Rows returned by a SELECT statement can be locked to prevent them from being changed by another transaction, by appending ''FOR UPDATE'' to the select statement.

There is no explicit table lock facility. You can prevent a table *schema* being modified, dropped or even reindexed using ''PROTECT TABLE'', but this does not include row-level modifications which are still allowed. ''UNPROTECT TABLE'' restores normal behavior.

No-Table Expression Select Syntax

It is not possible to select constant expressions. Only table fields can be selected.

Table Join Syntax

SearchServer does not really supports joins, but two different mechanisms are there to emulate at least some of the functionality. They all require planning ahead, though, since participating tables will have at least part of their schemas in common (column definitions).

First, you have a UNION clause for SELECT statements. Using UNION you can group different tables even on different servers. Tables must have all the columns in a query defined in a similar manner when created.

Second, you have ''views.'' With views, tables must be located on the same node and have the same schema. Only read-only access is granted with views, and they have to be described using a special syntax file. Please refer to the ''Data Administration and Preparation'' manual for more information on views, and ''SearchSQL Reference'' for a comparison between views and UNION.

Table and Column Names

Letters, numbers, and underscores (_) are valid characters in identifiers. The maximum size of table and column names is not known at this time.

SearchServer converts all identifiers to upper-case. Table and column names are not case sensitive. National characters can be used in identifier names.

Case Sensitivity of LIKE Operator

The LIKE operator is not case sensitive.

Row ID

The SearchServer ‘‘row id’’ pseudocolumn is called FT_CID and is of the INTEGER data type. FT_CID can be used in a WHERE clause, but only with the = operator.

Automatic Key or Sequence Generation

SearchServer does not support automatic key generation such as ‘‘auto increment’’ or ‘‘system generated’’ keys. The FT_CID, however, is not reissued when rows are deleted.

There is no support for sequence generators.

Automatic Row Numbering and Row Count Limiting

There is no pseudocolumn that sequentially numbers the rows fetched by a select statement.

Parameter Binding

Parameter binding is emulated by the driver. Both the `?` and `:1` style of placeholders are supported by the driver emulation.

The TYPE attribute to `bind_param()` is ignored, so no warning is generated for unsupported values.

Stored Procedures

There are no stored procedures or functions in SearchServer.

Table Metadata

`DBD::SearchServer` supports the `table_info()` method.

The COLUMNS ‘‘system table’’ contains detailed information about all columns of all the tables in the database, one row per column. You can tell if a row can contain NULLs via the NULLABLE column on the COLUMNS system table.

The COLUMNS system table uses the INDEX_MODE column to identify indexed columns and which indexing mode is used for them.

There are no keys in a SearchServer table.

Driver-specific Attributes and Methods

`DBD::SearchServer` has no driver-specific database handle attributes. It does have one driver-specific statement handle attribute:

ss_last_row_id

This attribute is read-only and is valid after an INSERT, DELETE or UPDATE statement. It will report the FT_CID (row ID) of the last affected row in the statement. You'll have to prepare/execute the statement (as opposed to simply doing it) in order to fetch the attribute.

There are no private methods.

Positioned updates and deletes

Positioned updates and deletes are supported using the ''WHERE CURRENT OF'' syntax. For example:

```
$dbh->do("UPDATE ... WHERE CURRENT OF $sth->{CursorName}");
```

Differences from the DBI Specification

None known.

URLs to More Database/Driver Specific Information

```
http://www.pcdocs.com
```

Concurrent use of Multiple Handles

`DBD::SearchServer` supports an unlimited number of concurrent database connections to the same server. It also supports the preparation and execution of a new statement handle while still fetching data from another statment handle associated with the same database handle.