

Chapter 3. SQL syntax

DB/C FS supports a subset of ANSI/ISO Standard SQL. DB/C FS supports the four basic SQL data manipulation statements (DML): SELECT, INSERT, DELETE and UPDATE. In addition, DB/C FS supports the following SQL data definition statements (DDL): ALTER, CREATE, and DROP.

Cursors are supported. Positioned DELETE and UPDATE are supported. Fetch is supported for obtaining rows of a result set. The cursor support is accomplished outside of the syntax of the SELECT, DELETE and UPDATE statements. Rather it is accomplished with additional parameters and functions in the protocol used to communicate between the client and the server. Client software varies in its approach to cursors. Some client software adds syntax to the SQL statements to support cursors. Other client software implements cursor specification outside of the SQL statement syntax.

SQL Data Manipulation Statements

The specific SQL data manipulation statements (DML) that DB/C FS supports are summarized with the following syntax information. Those elements enclosed in square brackets ([]) are optional.

```
SELECT [ DISTINCT ] select-list FROM table-list [ WHERE search-condition ]  
  [ ORDER BY sort-spec-list ] [ [ GROUP BY sort-spec-list ] [ HAVING having-search-condition ] ] |  
  [ [ FOR READ ONLY | FOR UPDATE [ OF column-list ] [ NOWAIT ] ] ]
```

```
UPDATE table-name SET column-set-value-list [ WHERE search-condition ]
```

```
DELETE FROM table-name [ WHERE search-condition ]
```

```
INSERT [ INTO ] table-name ( column-list ) VALUES ( value-list )
```

select-list is:

```
* | column-list
```

* refers to all columns of all tables of this query

column-list is a comma delimited list of one or more of these:

```
column-spec [ [AS] column-label ]  
table-name.*  
table-label.*  
string-literal [ [AS] column-label ]  
result-expression [ [AS] column-label ]
```

column-spec is:

```
column-name  
table-name.column-name  
table-label.column-name
```

*table-name.** and *table-label.** refer to all columns of the referenced table.

A *result-expression* is in algebraic form that contains constants, arithmetic operators, functions and columns. Functions consist of set functions, string functions, and CAST expressions.

Here is the list of set functions:

```
COUNT ( * )  
COUNT ( [ ALL | DISTINCT ] qualified-column-name )  
AVG ( [ ALL | DISTINCT ] qualified-column-name )  
MAX ( [ ALL | DISTINCT ] qualified-column-name )  
MIN ( [ ALL | DISTINCT ] qualified-column-name )  
SUM ( [ ALL | DISTINCT ] qualified-column-name )
```

In each of the set functions, **ALL** is the default. If **DISTINCT** is specified more than once in any expression in a **SELECT** statement, it must refer to the same column in all of the entries in this select statement that have **DISTINCT** specified.

The string value functions are:

```
UPPER ( value-expression )  
LOWER ( value-expression )  
value-expression || value-expression  
SUBSTRING ( value-expression FROM start [ FOR length ] )  
TRIM ( [ LEADING | TRAILING | BOTH ] [ char-expression ] FROM char-expression )
```

Note that '||' is the concatenate operator.

The CAST function is:

```
CAST ( value-expression AS datatype )
```

value-expression is:

```
column-name  
table-name.column-name  
table-label.column-name  
string-literal  
string value function
```

table-list is:

```
table-spec [ join-operator table-spec [ ON join-search ] ... ]
```

table-spec is:

```
table-name [ [ AS ] table-label ]
```

join-operator is one of:

```
,  
[ INNER ] JOIN  
LEFT OUTER JOIN
```

The comma operator and **INNER JOIN** are the same.

The **ON** *search-condition* clause is essentially the same as the **WHERE**, except that it can only contain operators and columns from the two tables being joined.

search-condition is an expression in algebraic form that contains constants, operators, string functions and columns from the table or tables specified in the statement. The operators are:

comparative operators:	= <> < > <= >= LIKE BETWEEN IN
logical operators:	AND OR NOT
arithmetic operators:	+ - * /

column-set-value-list is one or more of the following comma separated values:

column-name = *value*

sort-spec-list is one or more of these comma delimited values:

column-spec
column-label
result-set-column-number

having-search-condition is a *search-condition* which may optionally contain expressions using set functions.

column-list is a comma delimited list of *column-names* and *column-labels*.

column-name refers to a column defined in the DBD file. If a *column-label* is associated with a *column-name* by an **AS** clause, it refers to that column defined in the DBD file.

table-name refers to a table defined in the DBD file.

In a SELECT statement, a column name may be unqualified or qualified with a *table-name* or a *table-label* that refers to a *table-name*, which thus refers to a table defined in the DBD file.

value-list is a list of string and numeric values.

Sub-queries are not supported.

The use of **<>**, **NOT LIKE**, **NOT BETWEEN**, **NOT IN** and dissimilar **OR** conditions are not recommended on large files as indexes will not be utilized and the entire text file will be read sequentially.

The tables in a **JOIN** are processed from left to right. With this in mind, the tables should be ordered so that the left table(s) can provide data which can be used with the indexes of the right table(s). Doing this will enhance performance.

SQL Data Definition Statements

The specific SQL data definition statements (DDL) that DB/C FS supports are summarized with the following syntax information. Those elements enclosed in brackets ([]) are optional. Note that DDL must be used carefully. DB/C FS server processes do not periodically check the DBD file for changes. Therefore, DDL commands should only be used on tables that are not currently being used by other connections.

CREATE TABLE *table-name* (*column-definition*, . . .)

column-definition is:

column-name *datatype* [(*length*)]

Example:

CREATE TABLE TEST (ITEM CHAR(6), VEND CHAR(6), LIST NUMERIC(9,3), QYTO NUMERIC(6))