Open Source RDBMS for J2ME

jMeSQL version 0.1 supports the following SQL statements and syntax.

06.06.2007

Elizeu Nogueira da Rosa Jr.

http://www.jmesql.net

Alphabetical list:

ALTER TABLE <tablename> **ADD COLUMN** <columnname> Datatype [(columnSize[,precision])] [DEFAULT 'defaultValue' [NOT NULL]] [BEFORE <existingcolumn>];

Adds the column to the end of the column list. Optional attributes, size and default value (with or without NOT NULL) can be specified. The optional BEFORE <existingcolumn> can be used to specify the name of an existing column so that the new column is inserted in a position just before the <existingcolumn>. If NOT NULL is specified and the table is not empty, then a default value must be specified.

ALTER TABLE <tablename> **DROP COLUMN** <columnname>;

Drops the column from the table. Will not work if column is part of a primary key, unique or foreign key constraint.

ALTER TABLE <tablename> **ADD CONSTRAINT** <constraintname> **UNIQUE (**<column list>**);**

Adds a unique constraint to the table. This will not work if there is already a unique constraints covering exactly the same <column list>.

ALTER TABLE <tablename> **ADD CONSTRAINT** <constraintname> **FOREIGN KEY (**<column list>**) REFERENCES** <exptablename> **(**<column list>**)** [ON DELETE CASCADE];

Adds a foreign key constraint to the table, using the same constraint syntax as when the foreign key is specified in a table definition.

ALTER TABLE <tablename> **DROP CONSTRAINT** <constraintname>;

Drops a named unique or foreign key constraint from the table.

ALTER TABLE <tablename> **RENAME TO** <newname>;

ALTER INDEX <indexname> **RENAME TO** <newname>;

Names can be changed so long as they do not conflict with other user-defined or sytem-defined names.

CALL Expression ;

Any expression can be called like a stored procedure, including, but not only Java stored procedures or functions. This command returns a ResultSet with one column and one row (the result) just like a SELECT statement with one row and one column.

See also: Stored Procedures / Functions, Expression.

COMMIT [WORK] ;

Ends a transaction and makes the changes permanent.

See also: ROLLBACK, SET AUTOCOMMIT.

CONNECT USER username **PASSWORD** password ;

Connects to the database as a different user. Use "" for an empty password.

See also: GRANT, REVOKE

CREATE ALIAS function **FOR** javaFunction ;

Creates an alias for a Java function. The function must be accessible from the JVM in which the database runs. Example:
CREATE ALIAS ABS FOR "java.lang.Math.abs"

See also: CALL, Stored Procedures / Functions

CREATE [UNIQUE] **INDEX** index **ON** table (column [, ...]) ;

Creates an index on one or more columns in a table.
Creating an index on searched columns may improve performance.

See also: CREATE TABLE, DROP INDEX

CREATE [MEMORY | CACHED | TEMP | TEXT] **TABLE** name
(columnDefinition [, ...] [, constraintDefinition...]) ;

Creates a tables in the memory (default) or on disk and only cached in memory. Identity columns are autoincrement columns. They must be integer columns and are automatically primary key columns. The last inserted value into an identity column for a connection is available using the function IDENTITY(), for example (where Id is the identity column):
INSERT INTO Test (Id, Name) VALUES (NULL,'Test'); CALL IDENTITY();

columnDefinition:
columnname Datatype [(columnSize[,precision])] [DEFAULT 'defaultValue'] [[NOT] NULL] [IDENTITY] [PRIMARY KEY]

the default value must be enclosed in singlequotes

constraintDefinition:
[CONSTRAINT name]
UNIQUE (column [,column...]) |
PRIMARY KEY (column [,column...]) |
FOREIGN KEY (column [,column...]) REFERENCES refTable (column [,column...]) [ON DELETE CASCADE]

See also: DROP TABLE

CREATE TRIGGER name {BEFORE|AFTER} {INSERT|UPDATE|DELETE} **ON** table [FOR EACH ROW] [QUEUE n] [NOWAIT] **CALL** triggerClass ;

triggerClass is an application-supplied class that implements the org.jMeSQL.Trigger interface e.g. "myPackage.trigClass". It is the fire method of this class that is invoked when the trigger event occurs. Ensure that triggerClass is present in the classpath which you use to start jMeSQL.

When the 'fire' method is called, it is passed the following arguments:
fire (String name, String table, Object row[])
where 'row' represents the row acted on, with each column being a member of the array. The mapping of row classes to database types is specified in Datatypes

If the trigger method wants to access the database, it must establish its own JDBC connection. Note that this means any access is in a separate transaction. The jdbc:default:connection: URL is not currently supported.

Implementation note: In the interests of not blocking the database's main thread, each trigger runs in a thread that will wait for its firing event to occur; when this happens, the trigger's thread calls triggerClass.fire. There is a queue of events waiting to be run by each trigger thread. This is particularly useful for 'FOR EACH ROW' triggers, when a large number of trigger events occur in rapid succession, without the trigger thread getting a chance to run. If the queue becomes full, subsequent additions to it cause the database engine to suspend awaiting space in the queue. Take great care to avoid this situation if the trigger action involves accessing the database, as deadlock will occur. This can be avoided either by ensuring the QUEUE parameter makes a large enough queue, or by using the NOWAIT parameter, which causes a new trigger event to overwrite the most recent event in the queue. The default queue size is 1024. Note also that the timing of trigger method calls is not guaranteed, so applications should implement their own synchronization measures if necessary.

See also: DROP TRIGGER

CREATE USER username **PASSWORD** password [ADMIN] ;

Creates a new user or new administrator in this database. Empty password can be made using "".
Only an administrator do this.

See also: CONNECT, GRANT, REVOKE

CREATE VIEW viewname **AS SELECT ... FROM ...** [WHERE Expression] ;

A view can be thought of as either a virtual table or a stored query. The data accessible through a view is not stored in the database as a distinct object. What is stored in the database is a SELECT statement. The result set of the SELECT statement forms the virtual table returned by the view. A user can use this virtual table by referencing the view name in SQL statements the same way a table is referenced. A view is used to do any or all of these functions:

Restrict a user to specific rows in a table. For example, allow an employee to see only the rows recording his or her work in a labor-tracking table.

Restrict a user to specific columns. For example, allow employees who do not work in payroll to see the name, office, work phone, and department columns in an employee table, but do not allow them to see any columns with salary information or personal information.

Join columns from multiple tables so that they look like a single table.

Aggregate information instead of supplying details. For example, present the sum of a column, or the maximum or minimum value from a column.

Views are created by defining the SELECT statement that retrieves the data to be presented by the view. The data tables referenced by the SELECT statement are known as the base tables for the view. In this example, is a view that selects data from three base tables to present a virtual table of commonly needed data:

```
CREATE VIEW mealsjv AS
  SELECT m.mid mid, m.name name, t.mealtype mt, a.aid aid,
         a.gname + ' ' + a.sname author, m.description description,
         m.asof asof
    FROM meals m, mealtypes t, authors a
   WHERE m.mealtype = t.mealtype
               AND m.aid = a.aid;
```

You can then reference mealsjv in statements in the same way you would reference a table:

```
SELECT *
  FROM mealsjv
```

A view can reference another view. For example, mealsjv presents information that is useful for long discriptions that contain identifers, but a short list might be all a web page display needs. A view can be built that selects only specific mealsjv columns:

```
CREATE VIEW mealswebv AS
  SELECT name, author
    FROM mealsjv
```

See also: Expression, SELECT, DROP VIEW

DELETE FROM table [WHERE Expression] ;

Removes rows in a table.

See also: Expression, INSERT, SELECT

DISCONNECT ;

Closes this connection. It is not required to call this command when using the JDBC interface: it is called automatically when the connection is closed. After disconnecting, it is not possible to execute other queries (also not CONNECT) with this connection.

See also: CONNECT

DROP INDEX index ;

Removes the specified index from the database. Will not work if the index backs a UNIQUE of FOREIGN KEY constraint.

See also: CREATE INDEX

DROP TABLE table [IF EXISTS] ;

Removes a table, the data and indexes from the database. When IF EXIST is used, the statement returns true if the table does not exist.

See also: CREATE TABLE

DROP TRIGGER trigger ;

Removes a trigger from the database.

See also: CREATE TRIGGER

DROP USER username ;

Removes a user from the database.
Only an administrator do this.

See also: CREATE USER

DROP VIEW viewname [IF EXISTS] ;

Removes a view from the database. When IF EXIST is used, the statement returns true if the view does not exist.

See also: CREATE VIEW

GRANT { SELECT | DELETE | INSERT | UPDATE | ALL } [,...]
ON { table | CLASS "package.class" } **TO** { username | PUBLIC } ;

Assigns privileges to a user or to all users (PUBLIC) for a table or for a class. To allow a user to call a function from a class, the right ALL must be used. Examples:
GRANT SELECT ON Test TO GUEST
GRANT ALL ON CLASS "java.lang.String" TO PUBLIC
Only an administrator do this.

See also: REVOKE, CREATE USER

INSERT INTO table [(column [,...])]
{ VALUES(Expression [,...]) | SelectStatement } ;

Adds one or more new rows of data into a table.

REVOKE { SELECT | DELETE | INSERT | UPDATE | ALL } [,...]
ON { table | CLASS "package.class" } **TO** { username | PUBLIC } ;

Withdraws privileges from a user or for PUBLIC (all users) for a table or class.
Only an administrator may do this.

See also: GRANT

ROLLBACK [WORK] ;

Undoes changes made since the last COMMIT or ROLLBACK.

See also: COMMIT

Creates an SQL script describing the database. This file is saved on the machine where the database files are located.
Only an administrator may do this.

SELECT [{LIMIT n m | TOP m}][DISTINCT]
{ selectExpression | table.* | * } [, ...]
[INTO [CACHED | TEMP | TEXT] newTable]
FROM tableList
[WHERE Expression]
[GROUP BY Expression [, ...]]
[ORDER BY orderExpression [, ...]]
[{ UNION [ALL] | {MINUS|EXCEPT} | INTERSECT } selectStatement] ;

Retrieves information from one or more tables in the database.

tableList:
table [{ INNER | LEFT OUTER } JOIN table ON Expression] [, ...]

selectExpression:
{ Expression | COUNT(*) | {COUNT | MIN | MAX | SUM | AVG} ([DISTINCT] Expression) }

orderExpression:
{ columnNr | columnAlias | selectExpression } [ASC | DESC]

LIMIT n m: creates the result set for the SELECT statement first and then discards the first n rows and returns the first m rows of the remaining result set. Special cases: LIMIT 0 m is equivalent to TOP m or FIRST m in other RDBMS's; LIMIT n 0 discards the first n rows and returns the rest of the result set.

TOP m is equivalent to LIMIT 0 m

See also: INSERT, UPDATE, DELETE

SET AUTOCOMMIT { TRUE | FALSE } ;

Switches on or off the connection's auto-commit mode. If switched on, then all statements will be committed as individual transactions. Otherwise, the statements are grouped into transactions that are terminated by either COMMIT or ROLLBACK. By default, new connections are in auto-commit mode.

SET PASSWORD password ;

Changes the password of the currently connected user. Empty password can be set using ""

SHUTDOWN [IMMEDIATELY | COMPACT] ;

Closes the current database.

SHUTDOWN performs a checkpoint to creates a new .script file that has the minimum size and contains the data for memory tables only. It then backs up the .data file containing the CACHED TABLE data in zipped format to the .backup file and closes the database.

SHUTDOWN IMMEDIATELY just closes the database files (like an external poweroff); this command is used internally to test the recovery mechanism. This command should not be used as the routine method of closing the database.

SHUTDOWN COMPACT writes out a new .script file which contains the data for all the tables, including CACHED and TEXT tables. It then deletes the existing text table files and the .data file before rewriting them. After this, it backs up the .data file in the same way as normal SHUTDOWN. This operations shrinks all files to the minimum size.
Only an administrator may do this.

UPDATE table **SET** column = Expression [, ...] ;
[WHERE Expression]

Modifies data of a table in the database.

See also: SELECT, INSERT, DELETE

Datatypes: The types on the same line are equivalent.

NAME	RANGE	Java type
INTEGER | INT	as Java type	"int" | "java.lang.Integer"
VARCHAR	Integer.MAXVALUE	"java.lang.String"
CHAR | CHARACTER	Integer.MAXVALUE	"java.lang.String"
DATE	as Java type	"java.sql.Date"

The uppercase names are the data types names defined by the SQL standard or commonly used by RDMS's. The data types in quotes are the Java class names - if these type names are used then they must be enclosed in quotes because in Java names are case-sensitive. Range indicates the maximum size of the object that can be stored. Where Integer.MAXVALUE is stated, this is a theoretical limit and in practice the maximum size of a VARCHAR or BINARY object that can be stored is dictated by the amount of memory available. In practice, objects of up to a megabyte in size have been successfully used in production databases.

The recommended Java mapping for the JDBC datatype FLOAT is as a Java type "double". Because of the potential confusion it is recommended that DOUBLE is used instead of FLOAT.

VARCHAR_IGNORECASE is a special case-insensitive type of VARCHAR. This type is not portable.

In jMeSQL, when defining CHAR and VARCHAR columns, the SIZE argument is optional and defaults to 0. If any other size is specified, it is stored in the database definition but is not enforeced by default. Once you have created the database (before adding data), you can close the database and add a database property value:

```
sql.enforce_size=true
```

This will enforce the specified size and pad CHAR fields with spaces to fill the size.

CHAR and VARCHAR and LONGVARCHAR columns are by default compared and sorted according to POSIX standards. To use the current JRE locale for sorting and comparison, add the following database property to the properties file.

```
sql.compare_in_locale=true
```

Columns of the type OTHER or OBJECT contain the serialized form of a Java Object in binary format. To insert or update such columns, a binary format string (see below under Expression) should be used. Using PreparedStatements with JDBC automates this transformation.

Comments

```
-- SQL style line comment
// Java style line comment
/* C style line comment */
```

All these types of comments are ignored by the database.

Stored Procedures / Functions

Stored procedures are Java functions that are called directly from the SQL language or using an alias. Calling Java functions (directly or using the alias) requires that the Java class can be reached by the database (server). The syntax is:

```
"java.lang.Math.sqrt"(2.0)
```

This means the packacke must be provided, and the name must be written as one word, and inside " because otherwise it is converted to uppercase (and not found).

An alias can be created using the command CREATE ALIAS:

```
CREATE ALIAS SQRT FOR "java.lang.Math.sqrt"
```

When an alias is defined, then the function can be called additionally using this alias:

```
SQRT(2.0)
```

List of built-in functions and stored procedures

Numerical

COT(d) (returns the cotangent of an angle)
LOG10(d) (returns the logarithm (base 10))
MOD(a,b) (returns a modulo b)
PI() (returns pi (3.1415...))
ROUND(a,b) (rounds a to b digits after the decimal point)
SIGN(d) (returns -1 if d is smaller than 0, 0 if d==0 and 1 if d is bigger than 0)
TRUNCATE(a,b) (truncates a to b digits after the decimal point)
BITAND(a,b) (return a & b)
BITOR(a,b) (returns a | b)
ROUNDMAGIC(d) (solves rounding problems such as 3.11-3.1-0.01)

String

ASCII(s) (returns the ASCII code of the leftmost character of s)
CHAR(c) (returns a character that has the ASCII code c)
CONCAT(str1,str2) (returns str1 + str2)
DIFFERENCE(s1,s2) (returns the difference between the sound of s1 and s2)
HEXTORAW(s1) (returns translated string)
INSERT(s,start,len,s2) (returns a string where len number of characters beginning at start has been replaced by s2)
LCASE(s) (converts s to lower case)
LEFT(s,count) (returns the leftmost count of characters of s)
LENGTH(s) (returns the number of characters in s)
LOCATE(search,s,[start]) (returns the first index (1=left, 0=not found) where search is found in s, starting at start)
LTRIM(s) (removes all leading blanks in s)

RAWTOHEX(s1) (returns translated string)
REPEAT(s,count) (returns s repeated count times)
REPLACE(s,replace,s2) (replaces all occurrences of replace in s with s2)
RIGHT(s,count) (returns the rightmost count of characters of s)
RTRIM(s) (removes all trailing spaces)
SOUNDEX(s) (returns a four character code representing the sound of s)
SPACE(count) (returns a string consisting of count spaces)
SUBSTR(s,start[,len]) (alias for substring)
SUBSTRING(s,start[,len]) (returns the substring starting at start (1=left) with length len)
UCASE(s) (converts s to upper case)
LOWER(s) (converts s to lower case)
UPPER(s) (converts s to upper case)

Date / Time

CURDATE() (returns the current date)
CURTIME() (returns the current time)
DAYNAME(date) (returns the name of the day)
DAYOFMONTH(date) (returns the day of the month (1-31))
DAYOFWEEK(date) (returns the day of the week (1 means Sunday))
DAYOFYEAR(date) (returns the day of the year (1-366))
HOUR(time) (return the hour (0-23))
MINUTE(time) (returns the minute (0-59))
MONTH(date) (returns the month (1-12))
MONTHNAME(date) (returns the name of the month)
NOW() (returns the current date and time as a timestamp)
QUARTER(date) (returns the quarter (1-4))
SECOND(time) (returns the second (0-59))
WEEK(date) (returns the week of this year (1-53)
YEAR(date) (returns the year)

System / Connection

DATABASE() (returns the name of the database of this connection)
USER() (returns the user name of this connection)
IDENTITY() (returns the last identity values that was inserted by this connection)

System

IFNULL(exp,value) (if exp is null, value is returned else exp)
CASEWHEN(exp,v2,v2) (if exp is true, v1 is returned, else v2)
CONVERT(term,type) (converts exp to another data type)
CAST(term AS type) (converts exp to another data type)

See also: CREATE ALIAS, CALL

Expression:

[NOT] condition [{ OR | AND } condition]

condition:
{ value [|| value]
| value { = | < | <= | | = | < | != | IS [NOT] } value
| EXISTS(selectStatement)
| value BETWEEN value AND value
| value [NOT] IN ({value [, ...] | selectStatement })
| value [NOT] LIKE value [ESCAPE] value }

value:
[+ | -] { term [{ + | - | * | / } term]
| (condition)
| function ([parameter] [,...])
| selectStatement giving one value

term:
{ 'string' | number | floatingpoint
| [table.]column | TRUE | FALSE | NULL }

string:
Strings in jMeSQL are Unicode strings. A string starts and ends with a single ' (singlequote). In a string started with ' (singlequote) use '' (two singlequotes) to create a ' (singlequote).

The LIKE keyword uses '%' to match any (including 0) number of characters, and '_' to match exactly one character. To search for '%' itself, '\%' must be used, for '_' use '_'; or any other escaping character may be set using the ESCAPE clause.

name:
The character set for identifiers (names) in jMeSQL is Unicode.

A unquoted identifier (name) starts with a letter and is followed by any number of letters or digits. In unquoted identifiers, lowercase characters are converted to uppercase. Because of this, unquoted names are not case sensitive when used in SQL statements.

Quoted identifiers can be used as names (for tables, columns, constraints or indexes). Quoted identifiers start and end with " (one doublequote). A quoted identifier can contain any Unicode character, including space. In a quoted identifier use "" (two doublequotes) to create a " (one doublequote). With quoted identifiers it is possible to create mixed-case table and column names. Example: CREATE TABLE "Address" ("Nr" INTEGER,"Name" VARCHAR); SELECT "Nr", "Name" FROM "Address";

The equivalent quoted identifier can be used for an unquoted identifer by converting the identifier to all uppercase and quoting it. For example, if a table name is defined as Address2 (unquoted), it can be referred to by its quoted form, "ADDRESS2", as well as address2, aDDress2 and ADDRESS2. Quoted identifiers should not be confused with SQL strings.

Quoting can sometimes be used for identifiers when there is an ambiguity. For example:

SELECT COUNT(*) "COUNT" FROM MYTABLE;

Portability between different JRE locales is an issue when accented characters are used in unquoted identifiers. Because native Java methods are used to convert the identifier to uppercase, the result may vary among different locales. It is recommended that accented characters are used only in quoted identifiers.

When using JDBC methods that take table, column, or index identifiers as arguments, treat the names as they are registered in the database. With these methods, unquoted identifiers should be used in all-uppercase to get the correct result. Quoted identifiers should be used in the exact case combination as they were defined - no quote character should be included in the name. JDBC methods that return a result set containing such identifiers return unquoted identifiers as all-uppercase and quoted identifiers in the exact case they are registered in the database.

values:
A 'date' value starts and ends with ' (singlequote), the format is yyyy-mm-dd (see java.sql.Date).
A 'time' value starts and ends with ' (singlequote), the format is hh:mm:ss (see java.sql.Time).
A 'timestamp' or 'datetime' value starts and ends with ' (singlequote), the format is yyyy-mm-dd hh:mm:ss.SSSSSSSSS (see java.sql.Timestamp).

When specifying default values for date / time columns in CREATE TABLE statements, or in SELECT,INSERT, and UPDATE statements, special values 'now', 'today', 'current_timestamp', 'sysdate', 'current_time' and 'current_date' (case independent) can be used. 'now' is used for TIME and TIMESTAMP columns, 'today' is used for DATE columns. Example:

```
 CREATE TABLE T(D DATE DEFAULT 'today');
 CREATE TABLE T1(TS TIMESTAMP DEFAULT 'now');
```

Binary data starts and ends with ' (singlequote), the format is hexadecimal. '0004ff' for example is 3 bytes, first 0, second 4 and last 255 (0xff).

Any number of commands may be combined. With combined commands, ';' (semicolon) must be used at the end of each command to ensure data integrity, despite the fact that the program may not return an error when it is not used.

Conventions Used in this Document

[A] means A is optional
{ B | C } means either B or C must be used.
(and) are the actual characters '(' and ')' used in statements.
UPPERCASE words are keywords

This document is beta.
Based on original HSQLDB documentation. Updates by Elizeu Nogueira da R. Jr.