XEP-0043: Jabber Database Access

Justin Kirby
mailto:justin@openaether.org
xmpp:zion@openaether.org

2003-10-20
Version 0.2

Status	Type	Short Name
Retracted	Standards Track	

Expose RDBM systems directly to the jabber network

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Accessing a RDBMS in a generic fashion is a complex and difficult task. Consequently, this will not be an attempt to XMLize a generic Database API or query language. Instead, it will providing a simple mechanism for a JID to read/write data that it has access to and specifying a model for those schemas to use in xml.
This document has two aims.

1. Be able to request the available schemas
2. Perform near SQL-like data manipulation

Although designed for use with an RDBMS this document is not restricted to such uses. It may be used with any data storage system that can be broken down to a simple table, column/row format. for example comma delimited files.

2 Prerequisites

To understand the following sections of this document the reader must be aware of the following.

2.1 Namespace

The current namespace of http://openaether.org/projects/jabber_database.html will be used until this becomes a jep. Once officially accepted as a jep and approved as final by the council, it will become http://www.xmpp.org/extensions/xep-0043.html.

2.2 Elements

- version - specify the version of the protocol that the client/server supports
- database - specify the database/catalog has the following attributes:
  - name - name of the database/catalog
  - sql - embed native SQL queries directly
  - table - the element scopes the children into the table. has the following attributes:
    * name - name of the table
    * permission - what the user can do with the data
    * col - describes the column. has the following attributes
      · name - name of the column

        - type - SQL99 datatype of the column
        - size - size of the datatype if required
        - op - comparison operator, used only if child of where element
    - where - scopes col elements into a 'sql-like' where clause
        - col - see above
- proc - element scopes the children into a procedure has the following attributes:
    - name - name of the sproc
    - permission - what the user can do with the data
    - col - see above
    - result - indicated return value by running the procedure (restricted to integer)

2.3 Data Types

There are a limited subset of data types available:

- bit - a single 'bit', usually used to represent boolean values
- tinyint - signed 8bit integer, has a range from -128 to +127
- integer - signed 32bit integer, has a range from -2147483648 to +2147483647
- utinyint - unsigned 8bit integer, has a range from 0 to +255
- uinteger - usigned 32bit integer, has a range from 0 to +4294967296
- float - allowed values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38 (can NOT be unsigned)
- numeric - unlimited size (some databases constrain this though)
- date - resolution is one day. acceptable ranges vary (TODO: constrain minimal range to something)
- datetime - a date and time combination (often has range dependencies)
- timestamp - a datetime used most often to record events
- time - a time in the format HH:MM:SS (TODO: specify valid range)
- char - an unsigned byte representing a single character (ASCII)
- vchar - a variable width char
- text - an extremely large chunk of text
- blob - an extremely large chunk of binary data (encode in MIME)

2.4 Assumed Database Setup

All SQL/RDBMS units will be scoped in the xml hierarchy:

```
<database>
  <table>
    <col/>
  </table>
</database>
```

All examples will assume the existence of the following rdbms setup. A database named 'testdb' with tables created with following SQL script:

```
create table tbl_one
        (
          a_int int,
          a_float float,
          a_char  char(10)
        )
        create table tbl_two
        (
          a_date datetime,
      a_numeric numeric(9,3)
        )
```

3 Usage

3.1 Requesting Schemas

Listing 1: A simple schema request

```
<iq id="001" to="db.host" type="get">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
</iq>
```

This is a simple request to discover what tables/procedures exist on the database testdb. And what permissions are available to the user. All schema requests will respond within the scope that was asked for. This is to prevent unnecessary data from flooding the network. So the response for the above request would look something like:

Listing 2: Response to a schema request

```
<iq id="001" type="result" from="db.host">
  <database
```

```
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one" permission="both"/>
      <table  name="tbl_two" permission="read"/>
  </database>
</iq>
```

The response is scoped to only the 'children' of the request. Since the request was for the testdb database, only the tables within that database were returned in the result. The reason for the limitation is to prevent excessively large packets from filling the network from large schemas.
The response indicates that the user has both read and write permissions on the table 'tbl_one' and only read permissions on the table 'tbl_two'. Consequently, the user may only perform get requests on 'tbl_two'.

Listing 3: Request detailed table schema

```
<iq id="002" type="get" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one"/>
  </database>
</iq>
```

The response would look like:

Listing 4: Response to detailed request

```
<iq id="002" type="result" from="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one"    permission="both">
     <col name="a_int" type="int"/>
      <col name="a_float" type="float"/>
      <col name="a_char" type="char" size="10"/>
    </table>
  </database>
</iq>
```

The schema response for tbl_one is quite intuitive. Three columns exist, one called a_int of type int (integer), another a_float of type float and a third called a_char of type char with a size of ten characters.

3.2 Manipulating Data

Manipulation of data (select, insert, update, delete) will definitely not be elegant or easy. SQL allows for some fairly complex queries on any fully functional RDBMS. Consequently, the data manipulation will be relatively limited since it is not a goal to translate SQL into xml.

3.2.1 Selects

To indicate a select like query, specify an <iq> of type get. The table that the query is to be performed against must be specified. The columns that are to be returned in the result set must be scoped within the relative table. Any attribute on the <col> element besides name will be ignored. e.g. it is not required nor recommended to specify the data types or the sizes while performing a get.

Listing 5: Basic select

```
<iq id="003" type="get" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_int"/>
      <col name="a_float"/>
      <col name="a_char"/>
    </table>
  </database>
</iq>

      SQL Syntax:
        select a_int, a_float, a_char
        from tbl_one
```

It is also possible to specify a limit on the number of rows returned in the result set by specifying a value for the limit attribute.

Listing 6: Basic select with limit

```
<iq id="003" type="get" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one"  limit="2">
      <col name="a_int"/>
      <col name="a_float"/>
      <col name="a_char"/>
      </table>
  </database>
```

```
</iq>
```

In this case a limit of two rows will be returned in the result set.
The result set which is returned will contain all the rows that met the criteria of the select. There is no schema information beyond the column names included in the result set. Each 'row' in the result set is scoped within the corresponding <table> element. This allows for queries on multiple tables to be used in one <iq> packet.

Listing 7: Response to basic select

```
<iq id="003" type="result" from="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_int">1234</col>
      <col name="a_float">123.45</col>
      <col name="a_char">onetwothre</col>
    </table>
    <table  name="tbl_one">
      <col name="a_int">2345</col>
      <col name="a_float">234.56</col>
      <col name="a_char">twothreefo</col>
    </table>
  </database>
</iq>
```

3.2.2 Constraining Result Sets

It would be impractical to request the entire contents of the table every time you needed one row or a subset of the data. You can constrain the result set by specifying a where clause.

Listing 8: Select with constraints

```
<iq id="004" type="get" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_int"/>
      <col name="a_float"/>
      <col name="a_char"/>
      <where>
        <col name="a_int" op="eq">1234</col>
        <col name="a_float" op="lt" conj="and">200.00</col>
      </where>
    </table>
```

```
  </database>
</iq>

        SQL Syntax:
        select a_int, a_float, a_char from tbl_one
        where   a_int = 1234 and a_float < 200.00
```

Attributes only used in the <col> element within a <where> element are the op (for operator) and conj for (conjunction). The op is used for comparison operators such as <, >, =, <>, <=, >=

- eq - equivalent =
- neq - not-equivalent <>
- lt - less than <
- gt - greater than >
- let - less than or equivalent <=
- get - greater than or equivalent >=
- null - IS NULL (the column is null in the database sense of the word)

The conjuction attribute is used to combined constraints in the where clause

- not - to negate a result
- or - logical OR ||
- and - logical AND &&

Result

Listing 9: Response to select with constraints

```
<iq id="003" type="result" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_int">1234</col>
      <col name="a_float">123.45</col>
      <col name="a_char">onetwothre</col>
    </table>
  </database>
</iq>
```

3.2.3 Inserts

Inserting or altering the stored data in anyway requires setting the type attribute to a value of set. This indicates that the user wants to perform a 'insert/update'. The differentiating factor between an insert and an update operation is whether a <where> element is used. If there is no <where> element then it must be interpreted as an insert. If a <where> element does exist, then it must be interpreted as an update.

Listing 10: Inserting data

```
<iq id="004" type="set" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_int">3456</col>
      <col name="a_float">345.67</col>
      <col name="a_char">threefour</col>
    </table>
    <table  name="tbl_two">
      <col name="a_date">02/16/2002</col>
      <col name="a_numeric">123456789123.123</col>
    </table>
  </database>
</iq>

SQL syntax:
insert tbl_one (a_int, a_float,a_char)VALUES(3456,   345.67, '
   threefour')
insert tbl_two (a_date, a_numeric) VALUES('02/16/2002',
   123456789123.123)
```

Result
If there is no result set for the query, as in an update, insert, delete, then the response must indicate success or failure within the <table> element scope. An empty <table> element indicates success, and a <table> element containing an <error> element indicates a failure.

Listing 11: Response to data insert

```
<iq id="004" type="result" from="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one"/>
    <table name="tbl_two">
      <error code="380">permission denied on table</error>
    </table>
  </database>
```

```
</iq>
```

The insert into tbl_one succeeded since the response has an empty <table> element. However, the insert into tbl_two failed with a permission denied error. Which is indicated with a non-empty <table> element.

3.2.4 Updates

As stated previously, if the type attribute has a value of set and a <where> element exists, then it must be interpreted as an update.

Listing 12: Updating

```
<iq id="005" type="set" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table  name="tbl_one">
      <col name="a_char">aaaaaaaaaa</col>
      <where>
        <col name=c"a_int">1234</col>
      </where>
    </table>
  </database>
</iq>

        SQL Syntax:
    update tbl_one
    set a_char = 'aaaaaaaaaa'
    where a_int = 1234
```

Result
Again, if there is no result set returned by the query, then success or failure must be indicated.

Listing 13: Response to update

```
<iq id="005" type="result" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one"/>
  </database>
</iq>
```

3.2.5 Deletes

If the type attribute has a value of set and there are no <col> elements scoped within the <table> element, then the query must be interpreted as a delete.

Listing 14: Simple delete

```
<iq id="006" type="set"  to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one">
      <where>
        <col name="a_int" op="eq">1234</col>
      </where>
    </table>
    </database>
</iq>

      SQL Syntax:
      delete from tbl_one where a_int = 1234
```

Result
Again, if a result set is not generated by a query, then success or failure must be indicated by the <table> element

Listing 15: Response to delete

```
<iq id="006" type="result"    to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one"/>
  </database>
</iq>
```

3.3 Procedures

Procedures, or stored procedures [1] , are often handy to make frequently used sql queries execute faster. These are simply queries stored in a precompiled form and given a name with a list of parameters. Each RDBMS handles procedures differently, but the common characteristics are that they are stored server side and have in/out parameters.
The <proc> element will be used to indicate a procedure. It has similar characteristics to the <table> element. The core differences are that the <col> elements have permissions and a

[1] Apparently procedures are not as common in RDBMS as I thought. Postgres and MySQL have functions, but not procedures. So until I, or someone else, researches this issue this feature is on hold.

<result> element can be used to indicate the value returned by the procedure.
The permission attribute on a <col> element is used to indicate whether the parameter is in (read), out (write) or in/out (both).
The only result set acceptable from a procedure is that of the parameters or <col> element. If the procedure produces a result set outside of the parameters this should be ignored.

3.4 Errors

The server must be able to let the client know when an error occurs, instead of just being silent.

Code	Message	Description
399	Invalid Database Name	Returned when the client has requested information from a database which does not exist according to the component.
398	Invalid Table Name	Returned when the client has requested information from a table/procedure which does not exist according to the component.
397	Invalid Column Name	Returned when the client has requested information from a column which does not exist according to the component.
380	Permission Denied on Table	Returned when the requested action is not allowed for the user on the table
401	Access Denied	Returned when the user does not have permission to use the component.

If the user requests an action on a table which they do not have permission to do the following should be returned

Listing 16: Permission denied error

```
<iq id="004" type="error" from="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_two">
      <error code="380">permission denied on table</error>
    </table>
</database>
    </iq>
```

If the user is not allowed to access the component the following should be returned

Listing 17: General access denied

```
<iq id="004" type="error" from="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <error code="401">Access Denied</error>
  </database>
</iq>
```

3.5 Optional Features

There are requirements which can be provided by other jabber components/namespaces, namely the jabber:iq:browse namespace in-place of Version Negotiation. Due to the inherent limitations of the above data retrieval mechanisms more sophisticated querying techniques might be desired. The <query> element will extend the functionality

3.5.1 Embedded SQL

The abilities described in the Basics section are just that, basic. To provide more flexibility and allow for the full power of SQL without xmlifying everything, a <sql> element may be implemented to provide this feature.
The <sql> element must be scoped within the <database> element.

Listing 18: Embedded sql query

```
<iq id="007" type="get" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <sql> select a_int, a_float from tbl_one </sql>
  </database>
</iq>
```

Result

Listing 19: Response to embedded query

```
<iq id="007" type="result" to="db.host">
  <database
      name="testdb"
      xmlns="http://openaether.org/projects/jabber_database.html"/>
    <table name="tbl_one" permission="both">
      <col name="a_int" type="integer"/>
      <col name="a_float" type="float"/>
    </table>
```

```
    <table name="tbl_one">
      <col name="a_int">1234</col>
      <col name="a_float">123.45</col>
    </table>
    <table name="tbl_one">
      <col name="a_int">2345</col>
      <col name="a_float">234.56</col>
    </table>
  </database>
</iq>
```

Since SQL is so flexible, the result set schema is not known until it is returned as a result of the query. Consequently, it must be sent as the first 'row' of the returned result. Each following row will be the actual data queried for.
If multiple tables are used within one SQL statement, then then name attribute within the <table> element can not be accurately denoted with a single table name. The best way to deal with this situation is to simply use a unique identifier within the scope of the <database> element. This will allow for multiple <sql> results to be scoped within the same result.

3.5.2 Version Negotiation

It is expected that this protocol will grow and be extended to meet various demands. Therefore, version negotiation[2] will be incorporated up front.
When the connection initiator, client end-user or server/transport, starts a session, it must first send the version number it expects to use, otherwise, behavior is undefined.

```
<iq id="000" type="get" to="db.host">
  <database
      xmlns="http://openaether.org/projects/jabber_database.html">
      <version>0.1</version>
    </database>
</iq>
```

Three responses are possible from the server.

1. It supports that version number and responds with:

```
<iq id="000" type="result" from="db.host">
  <database
      xmlns="http://openaether.org/projects/jabber_database.html"
        >
```

[2] Version Negotiation is being killed since browsing, feature negotiation, or disco will be able to perform this function, however it might be useful as an optional feature for clients that don't implement these yet, especially considering none of these have been standardized.

```
    <version>0.1</version>
    </database>
</iq>
```

The type of 'result' indicates that the version request was successful and if the client is satisfied with the version number, may continue with schema requests or whatever.

2. It does not support that version number and responds with:

```
<iq id="000" type="error" from="db.host">
  <database
      xmlns="http://openaether.org/projects/jabber_database.html"
         />
</iq>
```

The type of 'error' indicates a failure in conforming to the desired version number. The server may optionally send an alternative option.

```
<iq id="000" type="error" from="db.host">
  <database
      xmlns="http://openaether.org/projects/jabber_database.html"
         >
      <version>0.2</version>
    </database>
</iq>
```

3. If the server has no idea what the client is talking about it should send the appropriate Jabber error code.

4 Limitations

1. No joins, roll ups, cubes
2. Views are not differentiated from tables
3. provides basic sql-like functionality only
4. Utilizes *lowest common denominator* approach

5 Todos

- define procedures; what they are and how they work
- determine value of adding administration features

6 Thanks

Thanks to Russell Davis (ukscone) for fine tuning the layout and wording of this jep. It would probably have been unreadable if it wasn't for him.

7 DTD and Schema

7.1 DTD

```
<!ELEMENT version (#PCDATA)>
<!ELEMENT error (#PCDATA)>
<!ELEMENT sql(#PCDATA)>
<!ELEMENT database (table | sproc | sql | error)*>
<!ELEMENT table (col | where | error)*>
<!ELEMENT where (col+)>
<!ELEMENT col (#PCDATA)>
<!ELEMENT proc(col | result | error)*>
<!ELEMENT result (#PCDATA)>
<!ATTLIST error code CDATA #IMPLIED>
<!ATTLIST database name CDATA #IMPLIED>
<!ATTLIST table
name CDATA #IMPLIED
    permission (read | write | both) #IMPLIED
    limit CDATA #IMPLIED
>
<!ATTLIST proc name CDATA #IMPLIED>
<!ATTLIST col
    name CDATA #IMPLIED
    size CDATA #IMPLIED
    op (eq | neq | lt | gt | let | get | null) #IMPLIED
    conj (not | or | and ) #IMPLIED
    permission (read | write | both) #IMPLIED
    type (bit | tinyint | integer | utinyint | uinteger |
          float | numeric | date | datetime | timestamp |
          time | char | vchar | text | blob) #IMPLIED
>
```

7.2 Schema

Anyone care to do this?