Using SQL tables in Web Security

SQL tables can be used with the web security tables through the use of active link tables. The web security code supports active link tables but does not actually use the active link table. Instead, it just creates appropriate SQL statements based on the table definitions and accesses the SQL tables directly. Therefore the speed running against SQL tables is approximately the same as against dbf tables and may actually be faster if there are a large number of users.

Alpha is currently preparing an automated process to create the SQL tables and corresponding active link tables. The suggestions here will use the same naming conventions for the tables and basic concepts.

Preparation

The security system requires four data tables. These must have a structure that is compatible with the web security. The SQL table names do not have to match the dbf table names, although they should be similar to allow easy identification. The SQL tables can have the exact same field names and structure or can have additional fields and different names. The dbf construction restricted field names to 10 characters. The SQL tables have no such restrictions, so more descriptive names could be used.

The field types must be compatible with the dbf types and meet the minimum lengths required. Unicode types may be used for the character fields, particularly for fields such as “SecQues” and SecAns” which may need Unicode support. All character fields can be variable length.

For example, the users table in dbf has a field named “guid” which is character and has a length of 36. A corresponding field in a SQL table (for example SQL Server) could be named “UserSecurityID” and be varchar(40). The dbf tables use logical fields which are not supported in all platforms. The corresponding SQL data type might be “bit” for SQL Server, or “tinyint( )” for MySQL where 0 = false and 1 = true.

Dbf Data Types

The dbf data types used are below with approximate equivalents

C = Character. Typically replaced with varchar() or nvarchar(). Nvarchar() may be used if the users may use a language for a security answers that need supported by unicode.

L = Logical. Typically replaced with bit or tinyint

T= Time. Usually the same as datetime
**SQL Tables to Create**

**User Table**

In the web security, the table is named “websecurity_users”. A suggested name for the SQL table would be “Users”. The table can be a member of any appropriate schema. This is the minimum construction required for the users security table. The format is “field name, dbf data type, length”.

```
guid,C,36
Userid,C,100
Password,C,60
RememberMe,L,1
RedirPage,C,60
PassEnt,T,17
PassExp,T,17
Email,C,100
SecQues,C,100
SecAns,C,60
updweb,T,17
updlocal,T,17
ulink,C,40
```

The “guid” or equivalent can be set as the primary key. Indexes may be created for “userid” and “ulink” if testing indicates a speed issue with large numbers of records. The expected automated process will add an index on “userid” as that is a common value used to find a user record.

Non-nullable fields

Guid
Userid

Defaults (suggested, not required as the security system will add value)

Guid = (replace(newid(),'','-'))  newid() is a SQL server function to create a standard guid (UUID) value. The field in the security table is misnames as it contains a modified guid value with the hyphens removed.

**Security Groups Table**

In the web security, the table is named “websecurity_groups”. A suggested name for the SQL table would be “Groups”. The table can be a member of any appropriate schema. This is the minimum construction required for the groups security table. The format is “field name, dbf data type, length”.

```
The “group_guid” or equivalent can be set as the primary key. No other indexes should be needed.

Non-nullable fields

Group_guid
Group_name

Defaults (suggested, not required as the security system will add value)

Group_guid = (replace(newid(),'','""))
newid() is a SQL server function to create a standard guid (UUID) value. The field in the security table contains a modified guid value with the hyphens removed.

Security Members table

In the web security, the table is named “websecurity_members”. A suggested name for the SQL table would be “Members”. The table can be a member of any appropriate schema. This is the minimum construction required for the groups security table. The format is “field name, dbf data type, length”.

Non-nullable fields

Group_guid
User_guid

“user_guid” is the field most often used for a search for records in this table. However, it is not unique and therefore not appropriate for a primary key. Another field could be added as an identity field for the primary key or a complex primary key could be created from user_guid + group_guid, which should be a unique value for each record. The expected automated process will add an index on “user_guid” as that is a common value used to find a user record.
**Security Expired table**

This table contains expired passwords and may or may not be used. It should be created anyway or an error will occur. In the web security, the table is named “websecurity_expired”. A suggested name for the SQL table would be “Expired”. The table can be a member of any appropriate schema. This is the minimum construction required for the expired security table. The format is “field name, dbf data type, length”.

```plaintext
user_Guid,C,36
Password,C,60
lastused,T,17
```

This table has the same issues for creating a primary key and the complex expression user_guid + password should be unique. The expected automated process will add an index on “user_guid” as that is a common value used to find a user record.

No fields should allow nulls.

**Foreign Keys**

No foreign key type constraints exist in the dbf format. However, an SQL format allows adding useful constraints. For example a foreign key can be added between members and users with the user “guid” as the parent.

**Active Link Tables**

After the SQL tables are constructed, active link tables can be defined. These must be named exactly the same as the dbf security tables and must be mapped to use the same field names as the dbf structure. The Alpha Five “Create Active-Link Tables” genie will create the tables in the database folder and not overwrite the existing security tables, which are in a web project folder.

The first step is to create a named connection to the SQL database that contains the new security tables. This may already exist

**Creating the User Table**

This is an example of the process to create a new active link user table. The same process can be used to build the other 3 tables. The example will use an actual table built in SQL Server as the source.
Step 1 – Open the genie

Step 2 – Make the Connection

Enter the name of the connection to the new security tables.

Step 3 – Select table and Fields

Select the name of the table created to contain security information for users. In this example, the table is in the “Security” schema and named “Users”. Select the fields that
correspond to the required security fields. They may have different and more descriptive names. They will be changed in field definition overrides. There may be additional fields in the SQL table, but do not include them in this selection. After selecting the fields by checking them, select the button at the bottom “Override Field Definitions”

Step 4 – Override Field Definitions

The fields in the SQL table may have different types and names that required by the web security system. The override allows setting the properties of the active link table to match the structure of the original dbf tables.
Select the field to edit, and double-click or press the button “Edit field definition”. This opens a small genie to set new values.

In this case, the SQL table has a field named “UserSecID” which corresponds to the “guid” field in the dbf table. Therefore, the name must have an override to rename it as “guid”.

In some cases, the data type may need changes as well. For example, a tinyint field may have been used in a MySQL table to correspond to the logical field “rememberme”. The override for “rememberme” may need the Data type changed here to “logical”.

You should not have to change the width. If the width in the SQL table is longer than in the original dbf definition, that will not cause any issues.

The goal is to have the override structure match the defined structure for the original dbf table as shown below. The data types must match the types used in the dbf, and the override name or default name must match the name in the dbf definition.
Step 5 – Set Optional Parameters

Strictly speaking, this is not necessary, but it is a good idea and the security code has a fallback option to actually use the active link table if a direct connection fails. Set the query location to “server-side”.

Override Field Definitions

You can override the definition, or name, of any of the fields in the Active-Link table. To override a field, double-click on a row.

<table>
<thead>
<tr>
<th>Default Name</th>
<th>Default Field Definition</th>
<th>Override Field Definition</th>
<th>Override Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usersecid</td>
<td>Character (60)</td>
<td></td>
<td>guid</td>
</tr>
<tr>
<td>UserID</td>
<td>Character (250)</td>
<td></td>
<td>UserID</td>
</tr>
<tr>
<td>Password</td>
<td>Character (60)</td>
<td></td>
<td>Password</td>
</tr>
<tr>
<td>Rememberme</td>
<td>Logical</td>
<td></td>
<td>Rememberme</td>
</tr>
<tr>
<td>Redirectpage</td>
<td>Character (60)</td>
<td></td>
<td>RedirectPage</td>
</tr>
<tr>
<td>Passwordexpires</td>
<td>Time</td>
<td></td>
<td>PasswordExp</td>
</tr>
<tr>
<td>EmailAddress</td>
<td>Character (250)</td>
<td></td>
<td>Email</td>
</tr>
<tr>
<td>Securityanswer</td>
<td>Character (60)</td>
<td></td>
<td>SecurityAnswer</td>
</tr>
<tr>
<td>UpdateWeb</td>
<td>Time</td>
<td></td>
<td>UpdateWeb</td>
</tr>
<tr>
<td>Securityquestion</td>
<td>Character (100)</td>
<td></td>
<td>SecurityQuestion</td>
</tr>
<tr>
<td>UpdateLocal</td>
<td>Time</td>
<td></td>
<td>UpdateLocal</td>
</tr>
</tbody>
</table>

Link Character (100)

Create Active-Link Tables

Specify Active-Link Table Options.

- Security
  - User/Password Prompting: None
- Queries
  - Prompt for query location: [ ]
  - Query location: Server-side
- Allow user to change preference: [ ]
- Miscellaneous
Step 6 – Save the table

The saved table name must match the expected name for the original dbf structure. In the case of the users table, this must be “websecurity_users.dbf”.

Other Tables

Repeat the process above for the other 3 security tables, “websecurity_groups”, “websecurity_members”, and “websecurity_expired”.

After construction, these files should exist in the database folder.

websecurity_expired.dbf  
websecurity_expired.dbP  
websecurity_expired.DDD  
websecurity_expired.DDM  
Websecurity_Expired.Ddx  
websecurity_groups.dbf  
websecurity_groups.dbP  
websecurity_groups.DDD  
websecurity_groups.DDM  
Websecurity_Groups.Ddx  
websecurity_members.dbf  
websecurity_members.dbP  
websecurity_members.DDD  
websecurity_members.DDM  
Websecurity_Members.Ddx  
websecurity_users.dbf  
websecurity_users.dbP
Migrating from DBF to SQL

If any existing security tables exist in the project folder, the data in those dbf tables can be moved to the new SQL tables by appending the data from each dbf table to the new active link table of the same name. It may be appropriate for an existing application to import the latest data to the desktop from a web server that contains the published web application. This will insure that the data added to the SQL database tables matches the data being used by the application.

The dbf table will be in the folder under the database folder named `\<database name>.WebProjects\<project name>.WebProject`.

Go to the operations tab and select a new operation. In the left column, select “Append records”. In the right column, select the web security table in the database folder as the target (master). The full path of any selection is shown below the windows, so you can see which table is selected. Use the append genie to find the “transaction table” or source. You can only see the full path of the table selected by actually selecting it, so you may get either the active link or original dbf on the first selection. The desired table is the original dbf, which is in the web projects folder.

Select append options that are appropriate. For example, both tables should have the same field names if the active link table was constructed properly. Since this append is typically only done once, you should select options to append all records, and probably should not save the append.

Repeat the append process to fill all of the active link versions of the tables. Although the appends are directed to the active link tables, the data will actually be saved in the source SQL database tables.
**Testing and Deployment**

After all data has been appended, you can delete all of the “websecurity_” files from the web project folder. It is a good idea to create a backup before deletion. Then move all of the “websecurity_” files in the database folder (these are the active link versions) to the web project folder.

You can open the “users and groups” desktop form to verify the data is correct. The data being viewed is the data being retrieved from the SQL database. If the data is correct, you can publish the web security data tables to the deployed web server. The web server will then access the same data on the SQL database using the same connection. There is no data actually stored in the dbf for an active link table. All of the data is pulled from the SQL database as required.

If the active link tables use the same connection as a deployed web application, the local view from the desktop “Users and Groups” actually views the same data as the web server. Therefore any change made locally will immediately show on the web side. The reverse is true as well. If data is changed by the web application, the local view will see the change on the desktop genie as soon as it is opened.

All existing web security functions should work correctly with the active link table method.