Database Schemas and Security

By Timothy Aaron Penner, Technical Services Team Member, 4D Inc.

Technical Note 08-43

Abstract

4D v11 SQL Release 3 introduces many improvements and features to the 4D v11 SQL line of products. One particularly useful feature is the support of *schemas* in 4D's integrated SQL engine. The implementation of this feature has resulted in modifications to the interface and the introduction of new SQL commands. This technical note describes the changes to the interface as well as the new SQL commands. A sample database component is included.

Introduction

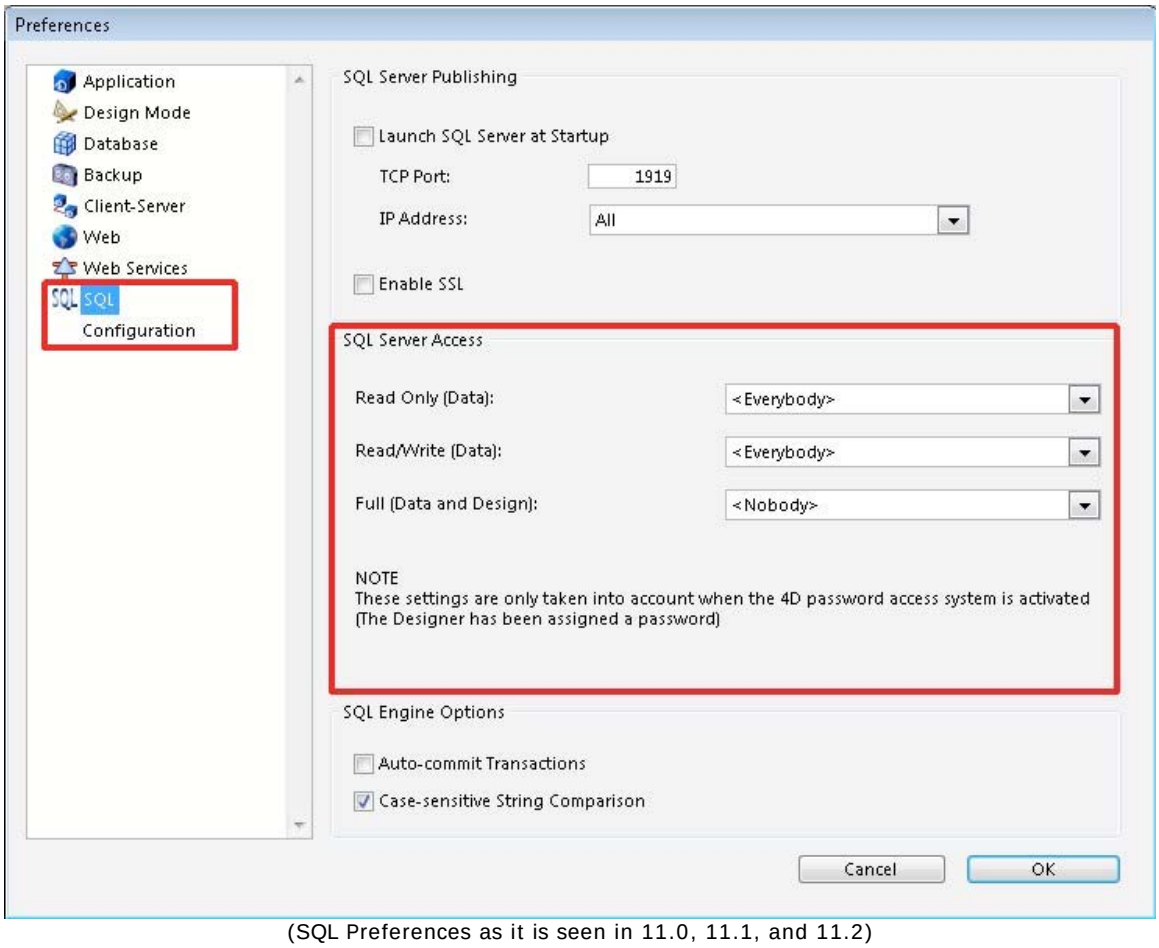
4D v11 SQL Release 3 introduces many improvements and features to the 4D v11 SQL line of products. One particularly useful feature is the support of *schemas* in 4D's integrated SQL engine. The implementation of this feature has resulted in modifications to the interface and the introduction of new SQL commands. This technical note describes the changes to the interface as well as how to use each of the new SQL commands. A comparison to the access rights available in previous versions of 4D v11 SQL is also covered in this technical note. A sample database component is included.

Schemas - What are they?

Schemas are a way of securing access to the tables in your database for external connections such as ODBC and SQL.

Comparison to previous versions (v11.0, v11.1, v11.2)

In previous versions of 4D v11, access rights for external connections were set globally for the database. This access was configured via the SQL preferences:



(SQL Preferences as it is seen in 11.0, 11.1, and 11.2)

From now on (v11.3), the access rights for external connections will be set per schema and each table can be configured to use one schema. The settings that are available in the SQL Preferences will now be used for the DEFAULT_SCHEMA which is automatically applied to all newly created tables as well as any tables that existed in the database upon upgrading to 11.3.

Each schema can have one 4D Group assigned each of the following access types:

- Read only (data)
- Read/Write (data)
- Full (data and structure)

NOTE: As in previous versions, access control only applies to external connections. The SQL code executed within 4D via the Begin SQL/End SQL tags, SQL EXECUTE, QUERY BY SQL, etc., still has full access.

Schemas - Why Use Them?

Schemas should be used in any database where people will be connecting through external means (i.e. via SQL or ODBC), and where data security is a concern. Schemas are easy to use, easy to set up, and get the job done.

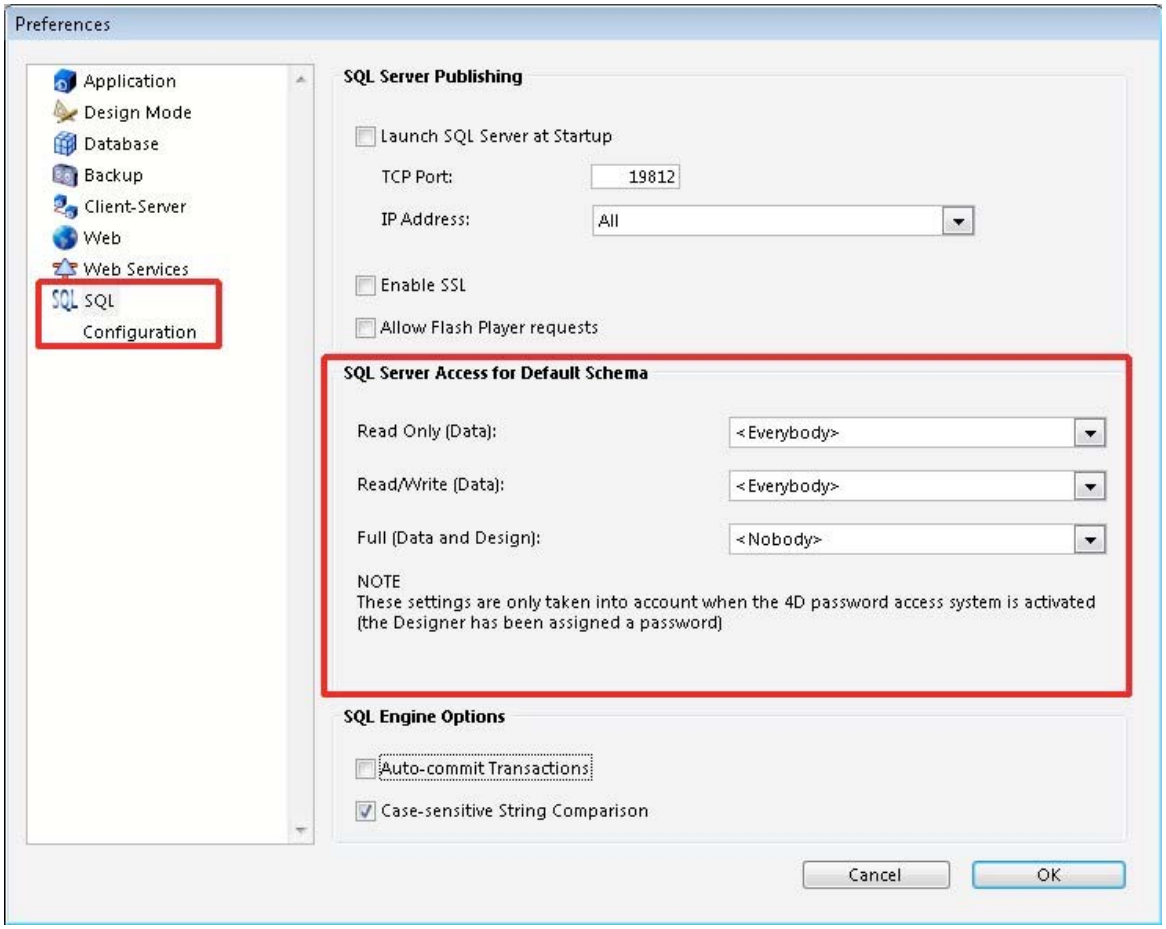
Schemas - How Do They Work?

A schema is a virtual object containing database tables. In SQL, the purpose of schemas is to assign specific access rights to different sets of database objects.

Schemas divide the database into independent entities which together make up the entire database. In other words, a table always belongs to one and only one schema.

When a database is created or converted with 4D v11 SQL Release 3 (or a subsequent version), a default schema is created in order to group together all the tables of the database. This schema is named "DEFAULT_SCHEMA." It cannot be deleted or renamed.

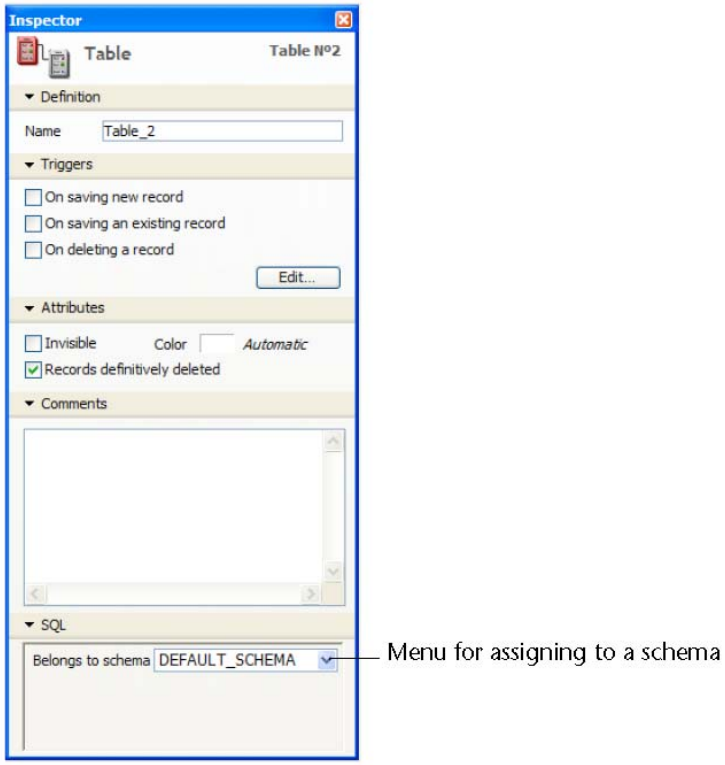
Modifications to the DEFAULT_SCHEMA can be carried out either through SQL code or the SQL page within the Preferences.



(SQL Preferences as it is seen in 11.3)

NOTE: Only the Designer and/or Administrator of the database can create, modify or delete schemas. If the access management system of 4D is not activated (in other words, if no password has been assigned to the Designer), all users can create and modify schemas with no restriction.

Schemas are created, modified and deleted via SQL commands. A new option in the Inspector palette can also be used to assign a table to a schema.



Creating a Schema

Creating a schema is accomplished with the following syntax:

```
BEGIN SQL
   CREATE SCHEMA mytest;
END SQL
```

The code snippet above will create a schema named mytest.

Modifying a Schema

Modifications to schemas are done programmatically through the use of SQL commands. The following sections outline specific syntax for various operations.

NOTE: The DEFAULT_SCHEMA can be modified either through code or through the SQL page within 4D's preferences.

Renaming a schema

Renaming a schema is accomplished with the following syntax:

```
BEGIN SQL
   ALTER SCHEMA mytest RENAME TO test;
END SQL
```

The code snippet above will rename the schema named mytest to test.

Granting READ Access Rights

Granting READ access to a schema is accomplished with the following syntax:

```
BEGIN SQL
   GRANT READ ON test TO TeamMembers;
END SQL
```

The code snippet above will grant READ access to the 4D group named TeamMembers on the schema named test.

NOTE: 4D allows group names to include spaces and/or accented characters that are not accepted by standard SQL. In this case, you must put the name between the [and] characters. For example: GRANT READ ON [schema] TO [admins!]

Granting READ_WRITE Access Rights

Granting READ_WRITE access to a schema is accomplished with the following syntax:

```
BEGIN SQL
   GRANT READ_WRITE ON test TO TeamLeaders;
END SQL
```

The code snippet above will grant READ_WRITE access to the 4D group named TeamLeaders on the schema named test.

Granting ALL Access Rights

Granting ALL access to a schema is accomplished with the following syntax:

```
BEGIN SQL
   GRANT ALL ON test TO Managers;
END SQL
```

The code snippet above will grant ALL access to the 4D group named Managers on the schema named test.

NOTE: In the current release of 4D v11 SQL, it is not possible to GRANT access to the EVERYBODY psuedogroup. The workaround for this is to create a 4D group containing all users, and then set the access right to that group. Newly created schemas also use the EVERYBODY psuedogroup for both READ and READ_WRITE access, so simply recreating the schema is another alternative.

Revoking Access Rights

Revoking access sets the group to NOBODY for the access level specified; this is accomplished with the following syntax:

```
BEGIN SQL
   REVOKE READ_WRITE ON test;
END SQL
```

The above code snippet will revoke READ_WRITE access for the schema named test; essentially setting READ_WRITE to NOBODY.

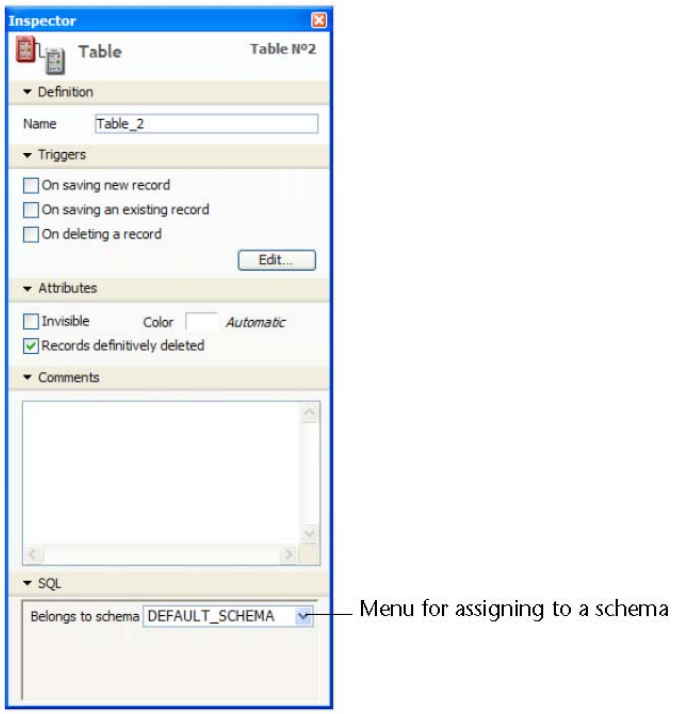
Applying Schemas

Schemas can be applied either via code or through the table Inspector on the Structure Editor. The syntax for applying a schema via code is:

```
BEGIN SQL
   ALTER TABLE [TABLE_1] SET SCHEMA test;
END SQL
```

The code snippet above will set the schema for [Table_1] to 'test'.

The following screenshot shows the new menu that has been added to the "Inspector" window of the Structure Editor:



Deleting a Schema

Deleting a schema is accomplished with the following syntax:

```
BEGIN SQL
   DROP SCHEMA test;
END SQL
```

The code snippet above will delete the schema named 'test'

Other ways of securing your database

In addition to the global access rights that were available in previous versions of 4D v11 SQL, the 'On SQL Authentication' database method that was introduced in 4D v11 SQL Release 2 could be used to trap the connection and do further authentication checks. This mechanism is still available in 4D v11 SQL Release 3.

Sample Database

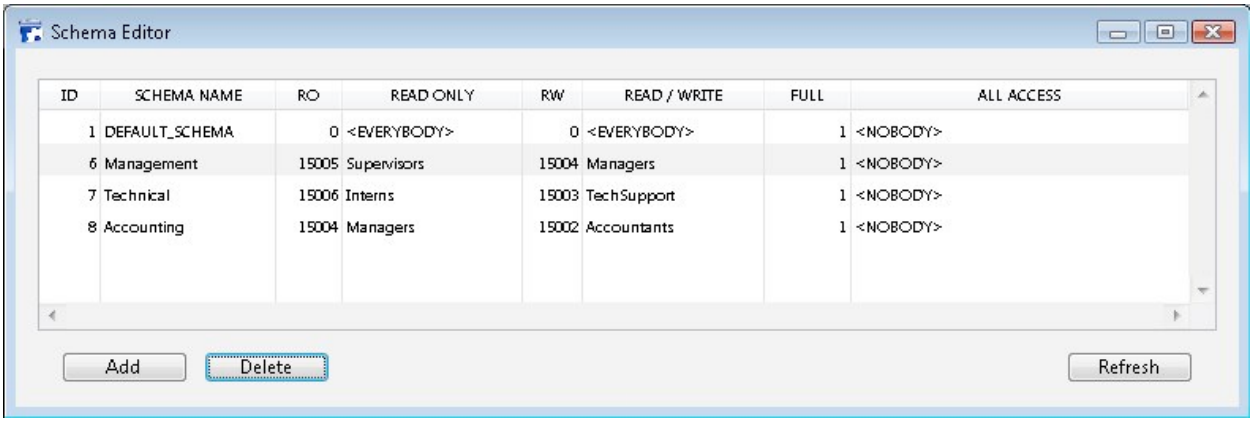
The sample database included with this technical note comes in two flavors:

- The source database of the component
- The componentized database

The componentized database can be dropped into an existing 4D v11 SQL Release 3 (or higher) project. It will allow you to more easily view and modify the schemas for the host database. The component method "SE_OPEN_SCHEMA_EDITOR" is used to open the editor.

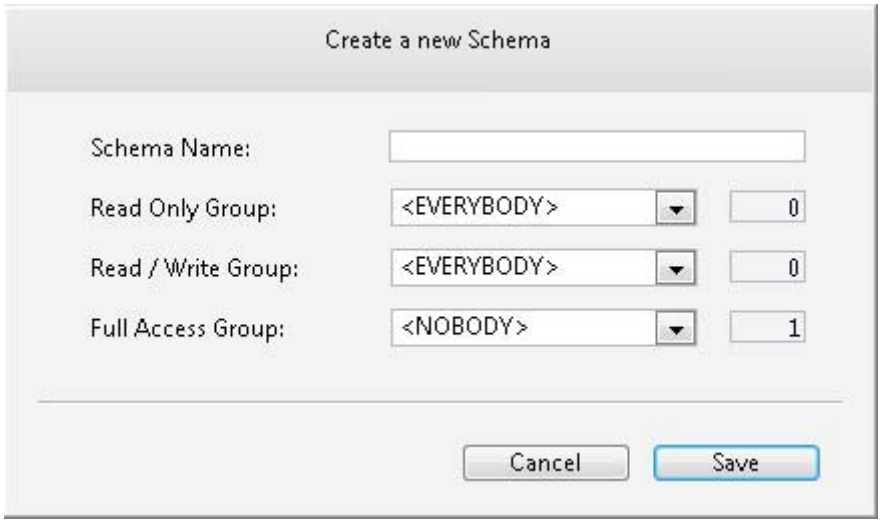
The following screenshot shows the schema editor open in a database that only has the DEFAULT_SCHEMA:

The following screenshot shows the schema editor open in a database that has a few user created schemas:



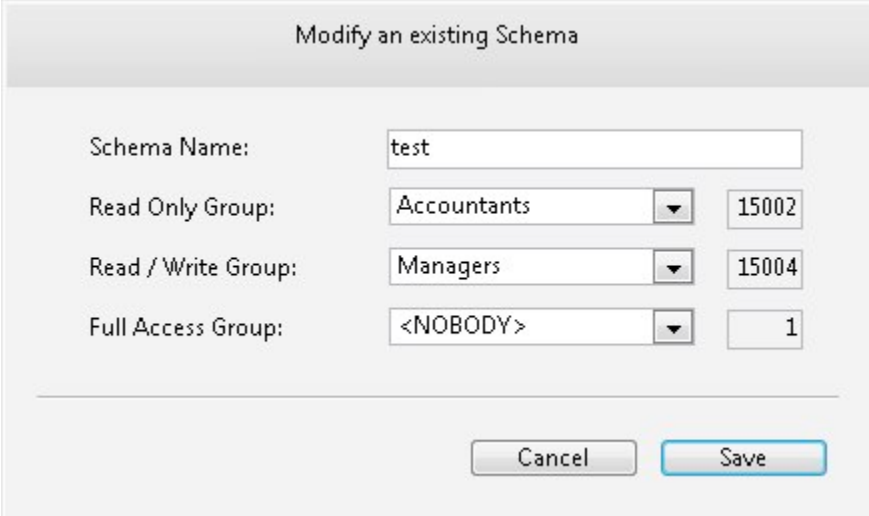
Adding a schema

Using the Schema Editor component you can add a schema via the GUI. To do so, simply click on the "Add" button from within the Schema Editor and the following dialog will appear:



Modifying a schema

Using the Schema Editor component you can modify a schema by double clicking on the schema you want to edit. Doing this will bring up the following dialog:



Caveats

It is not possible to GRANT access to the EVERYBODY psuedogroup in the current release of 4D v11 SQL therefore you will be prompted with a warning if you try to do this in the component. The workaround for this is to create a 4D group containing all users, and then set the access right to that group.

Newly created schemas also use the EVERYBODY psuedogroup for both READ and READ_WRITE access, so simply recreating the schema is another alternative. However deleting a schema will cause any tables that belonged to that schema to revert back to the DEFAULT_SCHEMA. You will then need to reset the schema for each table that was reverted.

Conclusion

This technical note described the general concepts of schemas in 4D v11 SQL Release 3 (or higher). Examples of how to create, modify, and delete schemas were presented. A sample database component is also presented as a means to offer an easier way for the developer to create and modify schemas. This information should allow the 4D v11 SQL developer to utilize schemas.