# Databases and Database Tables

Using an object of *Database* and *Database table* types allows to access the SQL database (via ODBC interface). An object of *Database* type defines:

| Parameter | Meaning |
|---|---|
| DSN | Name of defined Data Source Name. |
| Name | Name of the user with access rights to the database. |
| Password | User's password to the database. |

Such an object uniquely defines a database and a user, who will be used while working with the database. The parent and owner of objects of Database type is process D2000 DBManager (the suffix for object of *Process* type is *DBM*).
Note: starting with version 7.01.024 the object of Database type has an integer value equal to current number of connections. Details on connections are specified in the description of configuration dialog box of Database.

A child of an object of *Database* type is the object of *Database table* type. Such the object represents the table or view defined in the database. In its configuration, the object defines following parameters:

| Parameter | Meaning |
|---|---|
| Structure type | Object of Structure definition type. |
| Access | Access rights (*None, Read only, Modify*). |
| Table | Name of the table in the database. |
| Index | Name of the column in table that are regarded as a key item. |
| Optional | Name of columns in table whose existence will be optional (i.e. the columns from the structure definition that are not defined as optional, must be in the database table (they are required)). |
| Not Null | Name of columns in table whose value must be defined before inserting or editing in the database table (the operations to insert or modify the records of tables). |
| History depth | The parameter defines the name of a column of Absolute time Type and history depth as a number of months, days and hours. The parameter is optional. If the parameter is defined, corresponding process DbManager will automatically delete all the rows, values of which in defined columns is older than defined history depth. |

A table (in a database) is available to objects of Database table type. Process D2000 DBManager works exclusively with columns defined in a Structure definition (column names in structure definition must match those in database table). A database table can contain more or fewer columns than defined in structure definition. If a column is defined in an object of Structure definition and not in database table, invalid values will be assigned to this column during reading.

To detect, which columns are included in a database table, the ODBC function SQLColumns is used with input parameter (except others) such as a table name and a table owner's name. If a database table name contains a dot, D2000 DBManager will use the text before the dot as the user's name and the text behind the dot as the table name (e.g. the name of an object of Database *Table type* is *Jerry.MyTable*, so D2000 DBManager detects columns of the table named *MyTable* of the user *Jerry*). Otherwise process D2000 DBManager will use a name given in the configuration of the database, which is the parent of the database table.
The described logic is necessary due to existence of the databases, which contain more database tables with the same name assigned to various users.

When creating SQL commands, D2000 DBManager closes table and column names into the quotes. If the database is case sensitive, then the column names in the structure and table definition have to be the same (the same applies to the table names). For the PostgreSQL database, it is possible to disable using the quotes using the */NQ* parameter. Then it is not necessary for the table name (in object of the Table type) and the column names in the *Structure Definition* object to correspond exactly to the names of the appropriate tables and columns in the database.

After adding a column into structure definition in D200 system, D2000 DBManager calls the function SQLColumns and detects occurrence of the added column in the SQL table. If you add a new column into the table in SQL database, which is already included in the structure definition, it is possible to force the DBManager to call the function SQLColumns using the button Test.

The columns type equivalence must be ensured in the database table and *Structure definition* according to the following table:

| Structure item type | Database (SQL) type |
|---|---|
| Logical (Boolean) | Any integer type.<br>Value mapping: False - 0, True - 1, Oscillate - 2 |
| Integer | Any integer type. |
| Real | Nay real type. |
| Absolute time | TimeStamp |
| Relative time | Any real type. |
| Text | Fixed size array of characters. Usually Char(x) where x is a number of characters. |

| | |
|---|---|
| Object | Saving values for this type is not possible in principle. Therefore process DbManager always writes the number value of 0. Reading generates invalid values for the items |

**Note:** The function SQLColumns, which the process D2000 DBManager uses to query table columns in SQL database, is called:

- during the first access to table after the start of process D2000 DBManager,
- during the first access to table after saving the structure definition which is referenced in the configuration of the table,
- during the first access to table after saving the configuration of the table,
- after pressing the button Test in the configuration of the table.

Therefore if a new column is added to the table, the proper procedure is:

- adding column to the table in SQL database,
- adding column to the structure definition. If the column already existed in the structure definition, then save the table or press the button Test in the configuration of the table.
- accessing the table to verify the functionality (e.g. via browser).

If the structure definition is enhanced prior to adding some column to the table SQL database and in between the table has been accessed (so that the function SQLColumns had been called before the column was created), the newly created column will not be written to nor read from until the occurrence of one of the events which cause the call of function SQLColumns.

If the column is deleted from the table, the proper procedure is:

- renaming of the column in the structure definition e.g. to *Unused*
- renaming of the column in the table in SQL database
- accessing the table to verify the functionality (e.g. via browser)

If the column is renamed in the table in SQL database prior to renaming it in the structure definition, then the access to the table between these operations will result in error (because of referencing a non-existent column in SQL table).

**Related pages:**

Databases - configuration dialog box
Database tables - configuration dialog box