

# BECKHOFF

## TwinCAT 3 Database Server

### Uruchomienie i konfiguracja baz danych w środowisku TwinCAT 3

Wersja dokumentacji 3.0

Aktualizacja: 11.08.2022

Kontakt: *support@beckhoff.pl*

Beckhoff Automation Sp. z o. o.

## Spis treści

1 Wstęp .....	5
2 Tryb konfiguracji.....	6
2.1 Ustawienia serwera.....	6
2.2 Dodawanie bazy .....	7
2.3 SQL Query Editor .....	8
2.4 AutoLogGroup .....	12
2.4.1 AdsDevice.....	13
2.4.2 Symbols.....	13
2.4.3 DBTable.....	14
2.4.4 Uruchomienie .....	14
3 Tryb konfiguracji – przykład niestandardowej tabeli .....	16
3.1 Rzutowanie zmiennych .....	16
3.2 Niestandardowa grupa autologowania.....	17
4 Sterowanie grupami autologowania poprzez PLC.....	19
5 Tryb PLC Expert:.....	20
6 Tryb SQL Expert: .....	24
6.1 Wysłanie komendy SQL bez odczytu danych (np. INSERT / UPDATE).....	24
6.2 Wysłanie komendy SQL z odczytem danych (np. SELECT). .....	26

*Wszystkie obrazy są chronione prawem autorskim. Wykorzystywanie i przekazywanie osobom trzecim jest niedozwolone.*

*Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® i XTS® są zastrzeżonymi znakami towarowymi i licencjonowanymi przez Beckhoff Automation GmbH. Inne oznaczenia użyte w niniejszej prezentacji mogą być znakami towarowymi, których użycie przez osoby trzecie do własnych celów może naruszać prawa właścicieli.*

*Informacje przedstawione w tej prezentacji zawierają jedynie ogólne opisy lub cechy wydajności, które w przypadku rzeczywistego zastosowania nie zawsze mają zastosowanie zgodnie z opisem lub które mogą ulec zmianie w wyniku dalszego rozwoju produktów. Obowiązek przedstawienia odpowiednich cech istnieje tylko wtedy, gdy zostanie to wyraźnie uzgodnione w warunkach umowy.*

Uwaga! Poniższy dokument zawiera przykładowe zastosowanie produktu oraz zbiór zaleceń i dobrych praktyk. Służy on wyłącznie celom szkoleniowym i wymaga szeregu dalszych modyfikacji przed zastosowaniem w rzeczywistej aplikacji. Autor dokumentu nie ponosi żadnej odpowiedzialności za niewłaściwe wykorzystanie produktu. Dany dokument w żadnym stopniu nie zastępuje dokumentacji technicznej dostępnej online na stronie [infosys.beckhoff.com](http://infosys.beckhoff.com).

## 1 Wstęp

**TwinCAT Database Server** pozwala na komunikację TwinCAT'a z różnymi systemami bazodanowymi np. Microsoft SQL Server, MySQL itp. Przy prostych zadaniach wystarczy korzystać z graficznego konfiguratora serwera w TwinCAT. Konfigurator zapisuje konfigurację do pliku XML, który może być przesłany na urządzenie docelowe. Bardziej zaawansowane aplikacje mogą wymagać wykorzystania bloków funkcyjnych z biblioteki **Tc3\_Database**. TwinCAT Database Server umożliwia też korzystanie z komend SQL z poziomu programu PLC.

Konfiguracji TwinCAT Database Server można więc dokonać w trzech trybach:

- 1) Konfiguracji - tylko konfiguracja w TwinCAT, bez implementacji kodu PLC
  - 2) PLC Expert – realizacja komend SQL generowanych poprzez bloki funkcyjne
  - 3) SQL Expert – komendy SQL tworzone są w programie PLC i przesyłane są w całości do bazy
- Tryby można ze sobą mieszać.

W wielu przypadkach, oprócz konfiguracji TwinCAT Database Server, wymagana jest instalacja systemu zarządzania wybraną bazą danych. W tej instrukcji skorzystano z SQLite, która jest plikową bazą danych i nie wymaga instalacji dodatkowego oprogramowania.

Aby rozpocząć pracę z TwinCAT Database Server, konieczne jest zainstalowanie dodatku **Tc3\_DatabaseServer TF6420** na komputerze programisty oraz na komputerze, który będzie serwerem (jeśli nie będą to te same urządzenia). Dodatek **TF6420** dostępny jest pod adresem:

<https://www.beckhoff.com/pl-pl/products/automation/twincat/tfxxx-twincat-3-functions/tf6xxx-tc3-connectivity/tf6420.html>.

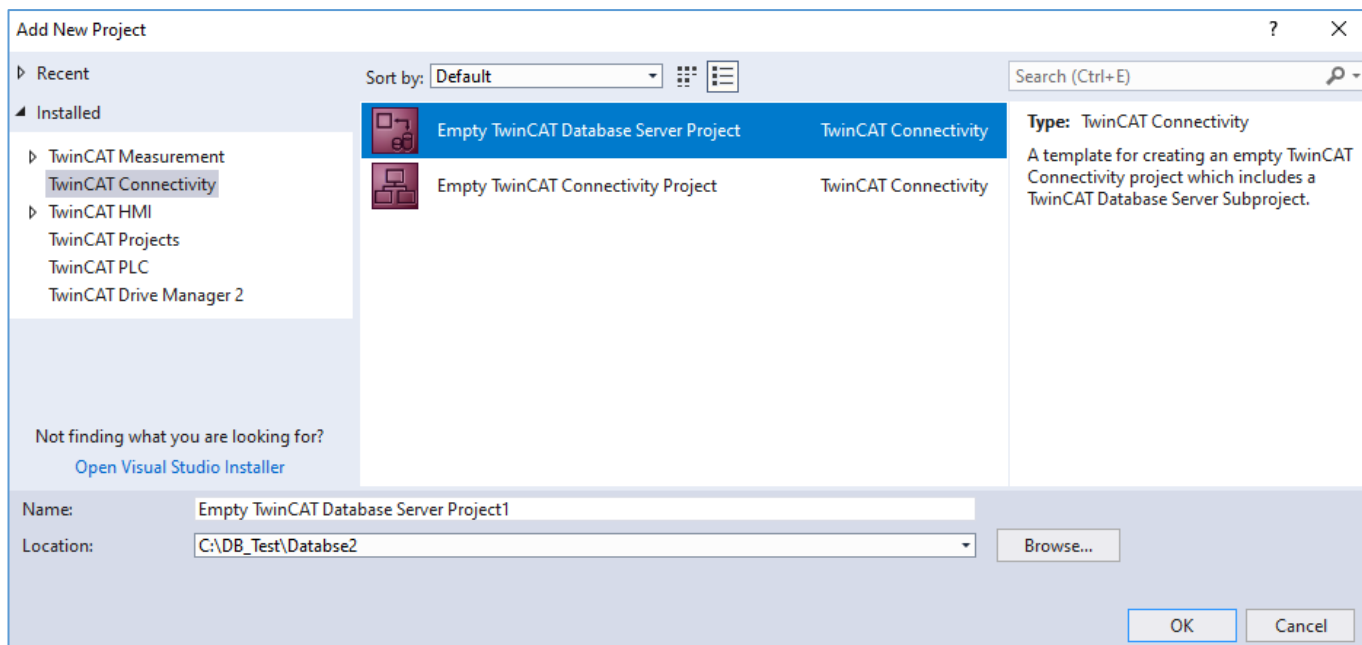
**Uwaga! Należy pamiętać o uruchamianiu instalatora Tc3\_DatabaseServer jako administrator.**

Po instalacji dodatku należy wygenerować licencję, na czas testów może być to licencja siedmiodniowa:

[https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420\\_tc3\\_database\\_server/262609675.html](https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420_tc3_database_server/262609675.html)

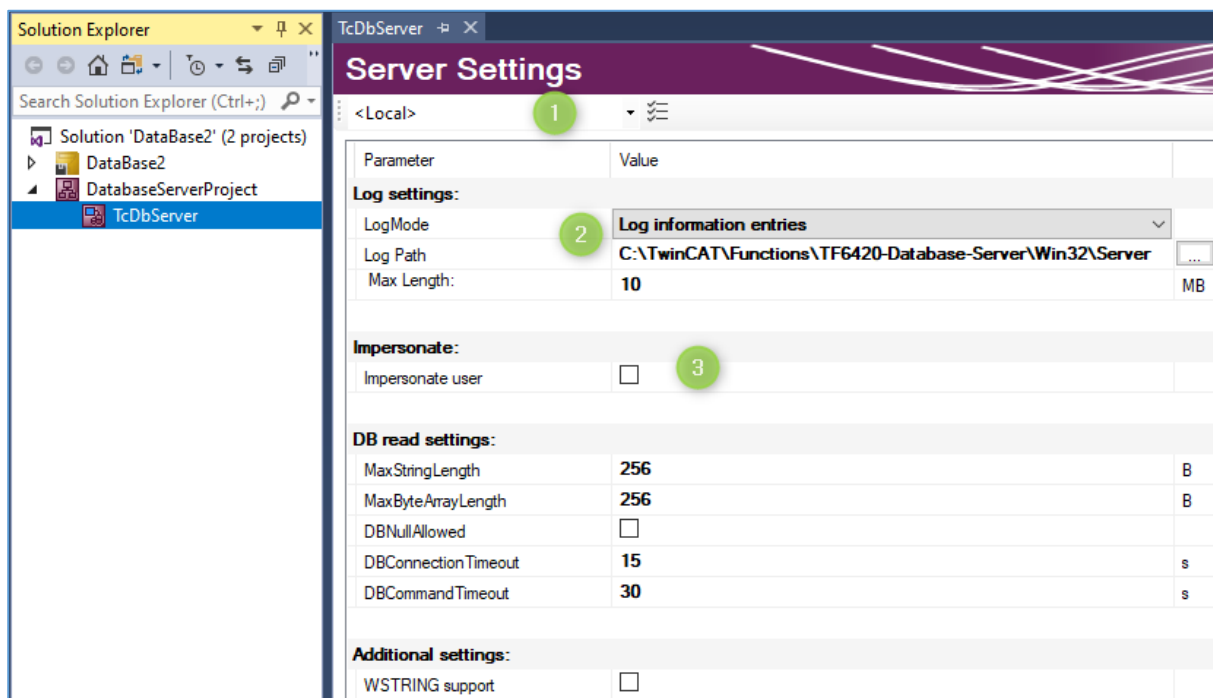
## 2 Tryb konfiguracji

Tryb konfiguracji umożliwia zestawienie połączenia z serwerem bazy danych (może być to serwer lokalny), utworzenie tabel w bazie oraz wybranie zmiennych z programu PLC, które mogą być automatycznie logowane do bazy. Aby wykonać powyższe kroki, należy w środowisku TwinCAT dodać nowy projekt **Empty TwinCAT Database Server Project**.



### 2.1 Ustawienia serwera

W pierwszym oknie wskazujemy urządzenie, które jest serwerem bazy danych (1), możemy również ustawić tryb i lokalizację zapisu logów (2) bazy (jest to rejestr błędów, który w przypadku problemów, pomoże nam namierzyć ich źródło).

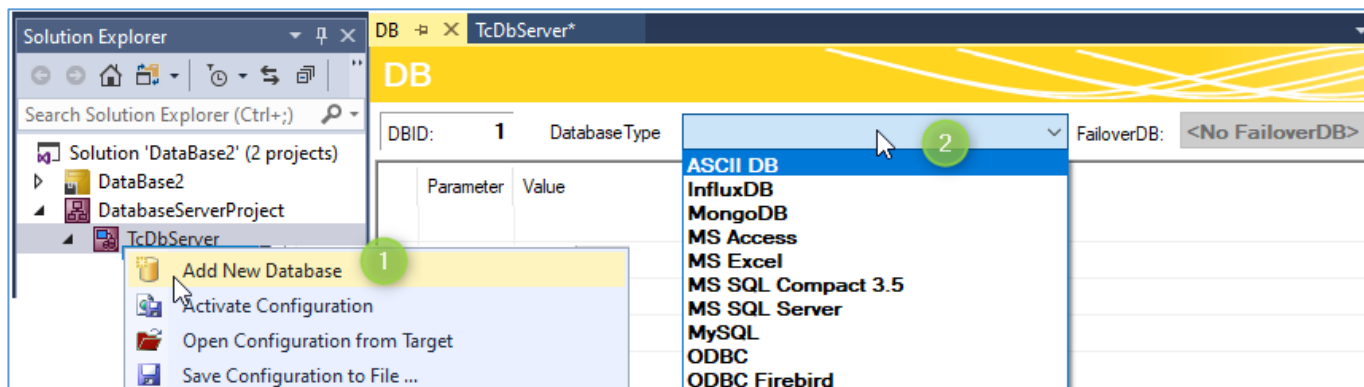


Opcja **Impersonate user** (3) jest zaznaczana, jeżeli jest potrzeba, żeby zrealizować połączenie przez sieć z plikową bazą danych taką jak Access lub SQL Compact.

Pozostałe ustawienia zmieniamy w razie potrzeby.

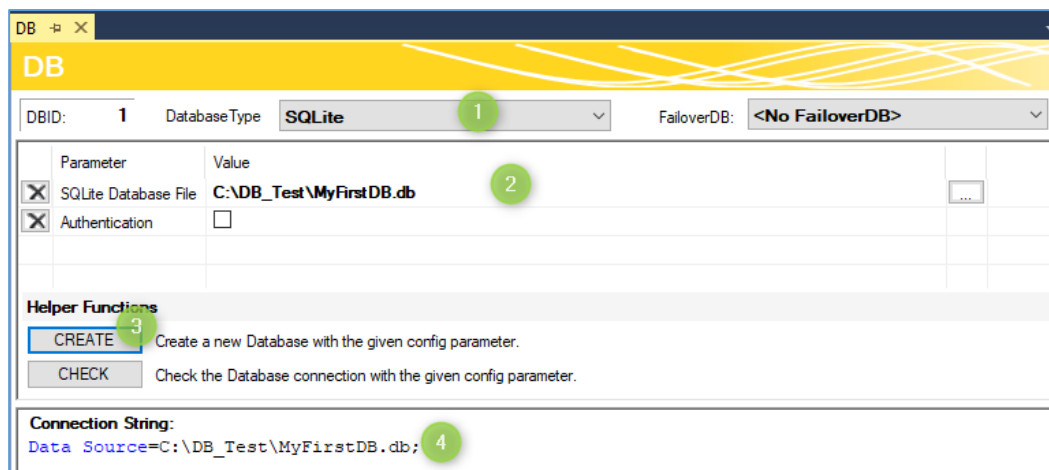
## 2.2 Dodawanie bazy

W celu dodania bazy należy kliknąć **PPM** na **TcDbServer** i wybrać **Add -> Add New Database** (1). Pojawi się okno do wskazania rodzaju bazy danych (2), z którą ma zachodzić komunikacja poprzez TwinCAT Database Server.



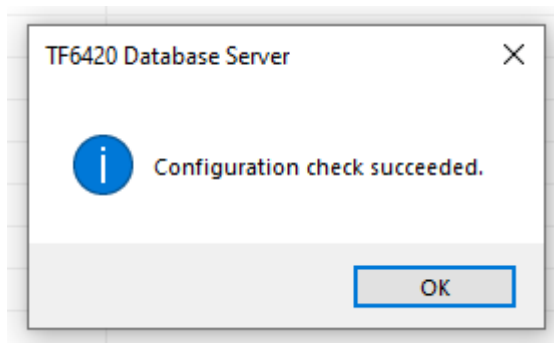
Z listy rozwijanej można wybrać jeden z obsługiwanych typów baz danych albo typ *ODBC*, gdzie jest możliwość ręcznego utworzenia *Connection String* do połączenia z bazami spoza listy (więcej informacji dostępne w dokumentacji), zgodnie ze standardem SQL.

DBID nadawane jest automatycznie przy każdej dodanej bazie i umożliwia rozróżnienie w konfiguracji i programie PLC wielu baz w jednym projekcie.



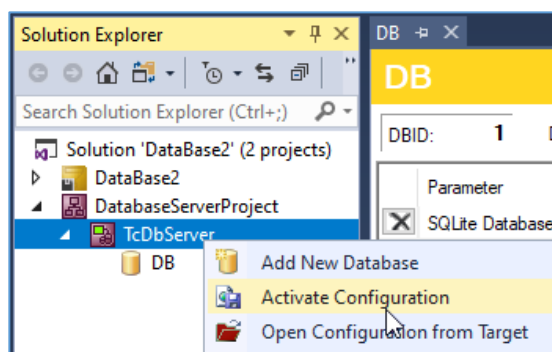
W przykładzie wybrano bazę **SQLite** (1). W polu **SQLite Database File** (2) należy podać ścieżkę do pliku bazy, o rozszerzeniu **.db** (plik nie musi na tym etapie istnieć). Opcjonalnie można podać hasło po zaznaczeniu opcji **Authentication**. Jeżeli baza ta jeszcze nie istnieje, można ją utworzyć klikając **Create** (3). Wybierając **Check** można sprawdzić, czy połączenie z bazą działa. W tym przypadku, po kliknięciu przycisku **Check**, powinien pojawić się komunikat pokazany obok.

Na dole okna widać automatycznie wygenerowany **Connection String** (4) zawierający wykonane ustawienia.

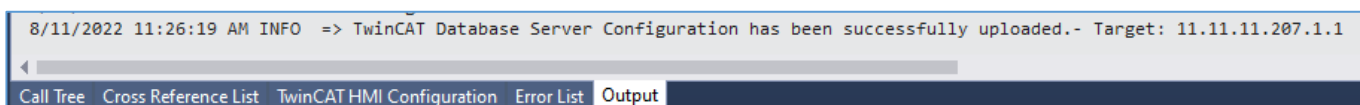


Po tak wykonanej konfiguracji, można dodatkowo wybrać w polu **FailoverDB** tak zwaną failover database, w której zachowane zostaną dane w razie napotkania błędu w trybie konfiguracyjnym. W razie rozłączenia z siecią, ta funkcja automatycznie zapewni, że dane nie przepadną i zostaną zapisane w innym miejscu.

Na zakończenie tego etapu należy zapisać zmiany i aktywować konfigurację projektu bazy danych (1) **(nie mylić z aktywowaniem konfiguracji PLC)**. Spowoduje to przesłanie pliku z konfiguracją na urządzenie docelowe wybrane w *Server Settings*.

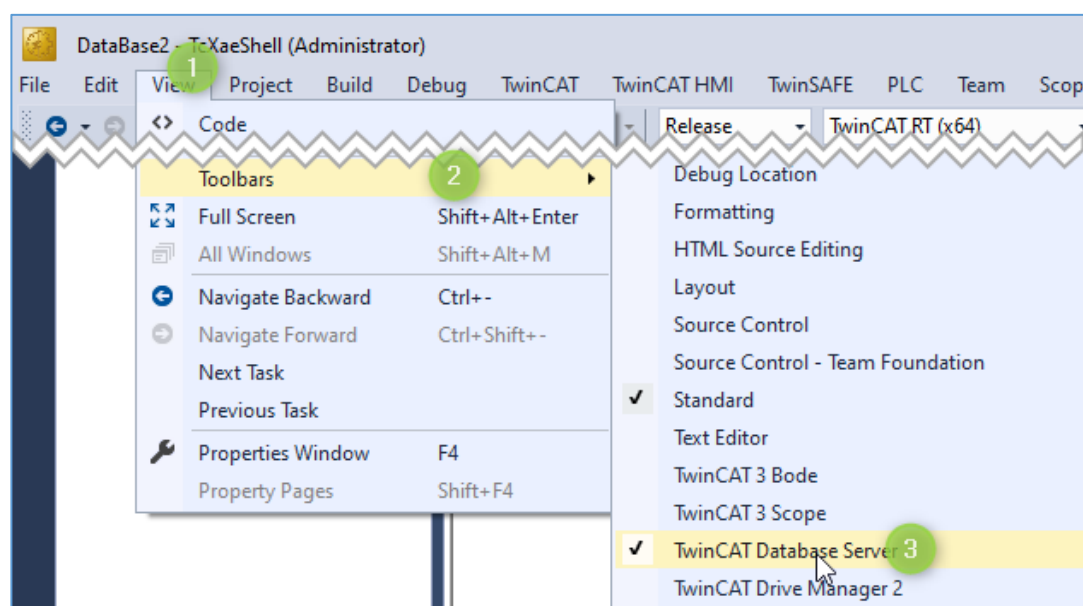


Operacja ta nie wywołuje żadnego okna popup, należy obserwować okno błędów oraz okno **Output** (Ctrl+Alt+O). Jeśli aktywacja się powiodła, powinna pojawić się informacja jak poniżej:



## 2.3 SQL Query Editor

Sql Query Editor służy do ręcznego tworzenia/usuwania tabel oraz wpisów. Aby otworzyć okno edytora należy w pierwszej kolejności uaktywnić Toolbar dodatku Database Server:

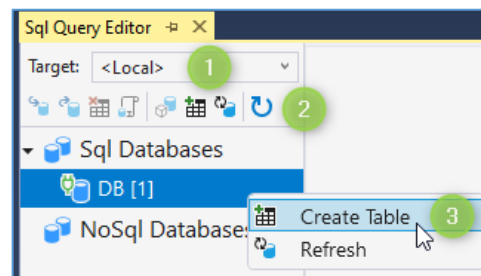




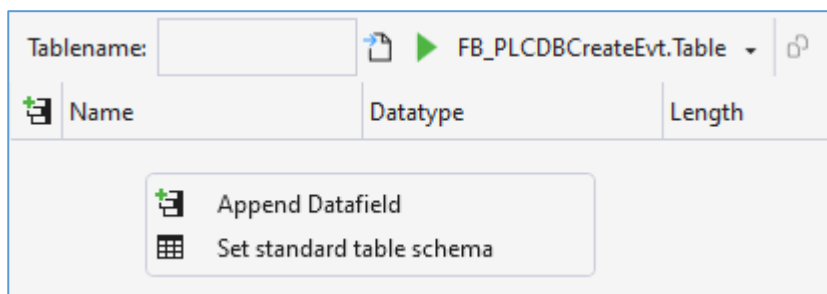
Z dodanego Toolbara należy wybrać ikonę **Sql Query Editor**.



W oknie jak obok należy wybrać serwer bazy danych (1), w razie potrzeby odświeżyć listę dostępnych baz (2) i na danej bazie wybrać komendę **Create Table**.

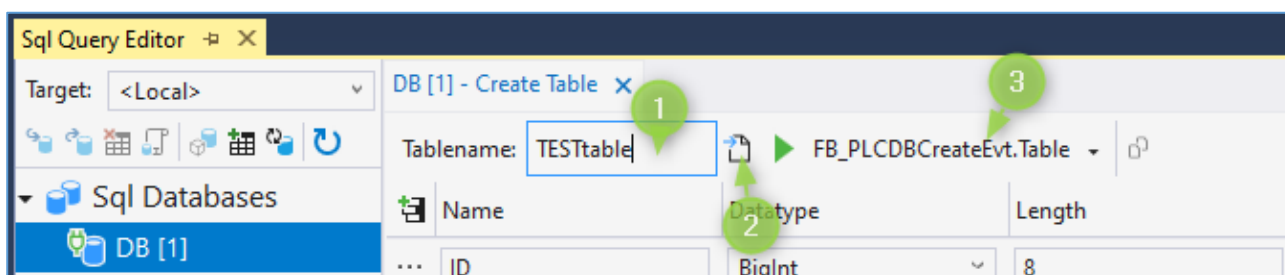


W przykładzie wykorzystano standardową strukturę tabeli. Aby utworzyć taką tabelę należy na pustym polu kliknąć **PPM** i wybrać opcję **Set Standard Table Schema**.

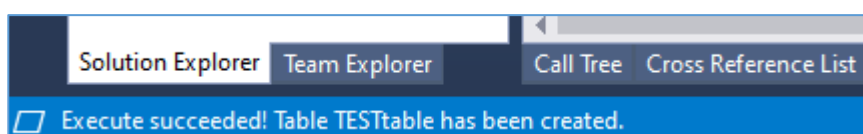


Kolumny zawierają kolejno: auto inkrementujące się ID wiersza, będące kluczem głównym; stempel czasowy zawierający aktualną datę i godzinę; nazwę zmiennej; wartość zmiennej. Pozostawienie ich bez zmian pozwoli na wykorzystanie standardowej metody logowania tabeli.

W kolejnym kroku należy wpisać nazwę tabeli (1), automatycznie wygenerować komendę SQL (2) i ją wykonać (3).

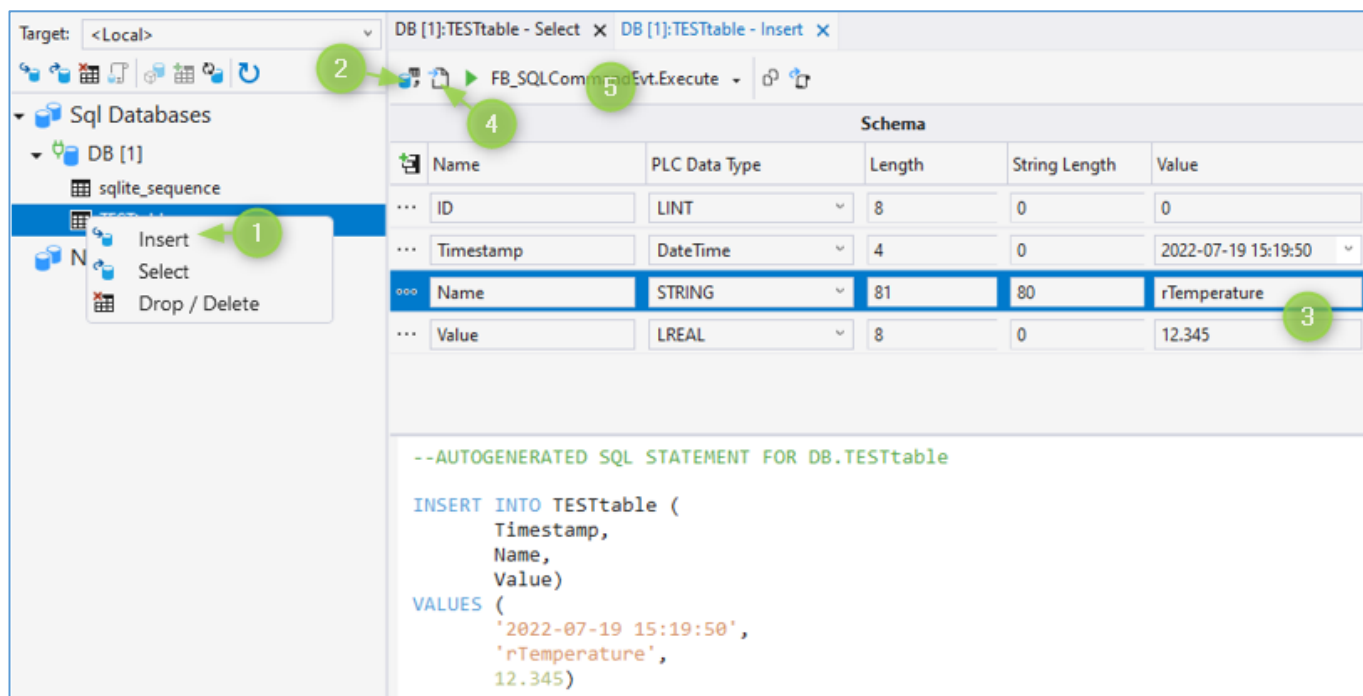


Jeżeli operacja się powiodła, na dolnym pasku statusowym środowiska TwinCAT, powinna się pojawić informacja:

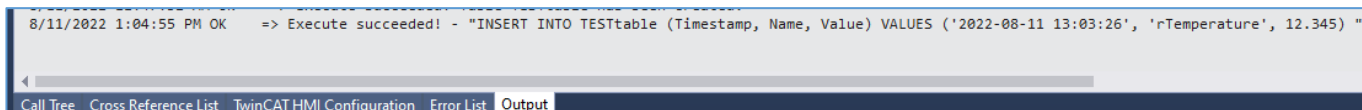


Aby ręcznie wykonać wpis do bazy (w celach testowych) należy:

- kliknąć na utworzoną wcześniej tabelę **PPM** i wybrać **Insert** (1)
- wybrać opcję **Get Table Schema** (2) (powoduje automatyczne nadanie wartości Timestamp)
- z prawej strony w kolumnie **Value** wpisać wartości, które mają trafić do bazy (3)
- kliknąć komendę **Create Query** (4) (generowanie wyrażenia SQL)
- wykonać komendę opcją **FB\_SQLCommandEvt.Execute** (5)

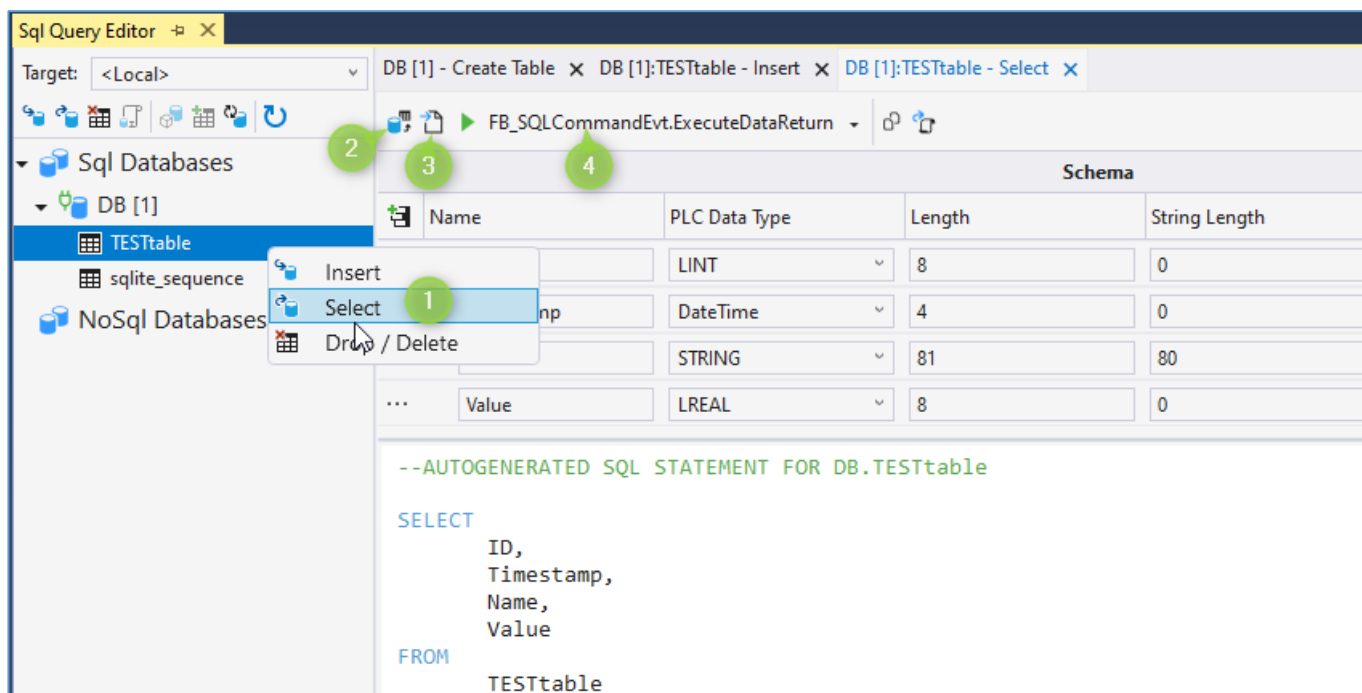


Ponownie w oknie **Output** powinna pojawić się informacja:



Aby odczytać wszystkie wartości znajdujące się w tabeli należy:

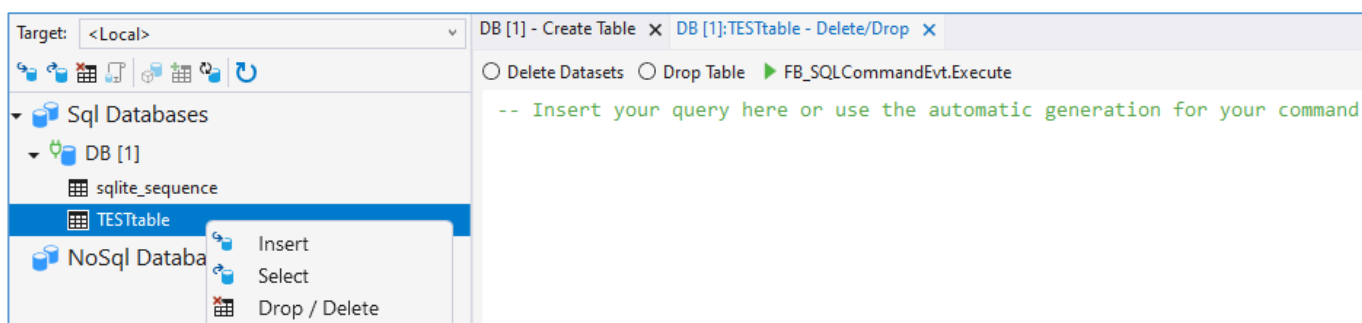
- kliknąć **PPM** na tabelę i wybierać opcję **Select** (1)
- wybrać kolejno opcję **Get Table Schema** (2) oraz **Create Query** (3) (utworzy się zapytanie do bazy o wszystkie dane z tabeli)
- wykonać komendę opcją **FB\_SQLCommandEvt.ExecuteDataReturn** (4)



Aby wczytać tylko część wyników należałoby wykorzystać komendę WHERE zgodnie ze standardem języka SQL. Można również ręcznie wpisać całą komendę SQL, której wyniki wpisane zostaną do tabeli w dolnej części okna, trzeba jednak zadbać o zgodność typów zmiennych. Informacje o równoważnych typach zmiennych znajdują się pod linkiem:

[https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420\\_tc3\\_database\\_server/27021597872201739.html](https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420_tc3_database_server/27021597872201739.html)

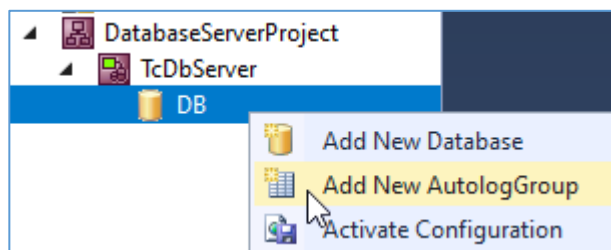
Ostatnia opcja - **Drop/Delete** - służy do usuwania wszystkich wpisów z tabeli (**Delete Datasets**) lub całej tabeli (**Drop Table**). Aby usunąć tylko wybrane elementy należy ręcznie wpisać komendy zgodnie ze standardem języka SQL.



## 2.4 AutoLogGroup

Docelowym krokiem trybu konfiguracyjnego jest dodanie i skonfigurowanie grupy autologowania. Grupa pozwala na automatyczne przekazywanie zmiennych z PLC do bazy albo na odwrót. Do jednej bazy danych może być dołączonych wiele różnych grup autologowania jednocześnie.

Aby rozpocząć pracę z grupami należy dodać je na wybranej bazie, klikając **PPM** i wybierając opcję **Add New AutologGroup**:



Pierwsze okno, które się pojawi, to główne ustawienia trybu logowania:

- **Start Up** – wybranie opcji *Automatic* spowoduje uruchamianie logowania grupy razem ze startem TwinCATA, opcja *Manual* wymaga sterownia grupą z poziomu kodu PLC

- **Direction** - decyduje o kierunku przepływu danych. Wybranie *DeviceAsTarget* daje możliwość uaktualniania zmiennych PLC wartościami znajdującymi się w bazie. Wybranie *DeviceAsSource* pozwoli na zapisywanie zmiennych PLC na jeden z trzech sposobów, które konfiguruje się w polu *Write Mode*

- **Write Mode:**

- *UPDATE* – wartości zmiennych w bazie będą aktualizowane (nadpisywane)

- *APPEND* – wartości będą każdorazowo dopisane jako nowy wiersz

- *RINGBUFFER\_TIME* – określa maksymalny czas żywotności rekordu w sekundach, po upływie tego czasu zostanie on usunięty. Czas określany jest w polu *Ringbuffer Parameter*

- *RINGBUFFER\_COUNT* – określa maksymalną ilość rekordów, po jej przekroczeniu kolejne rekordy będą usuwane, zaczynając od najstarszego. Ilość rekordów jest określana w polu *Ringbuffer Parameter*

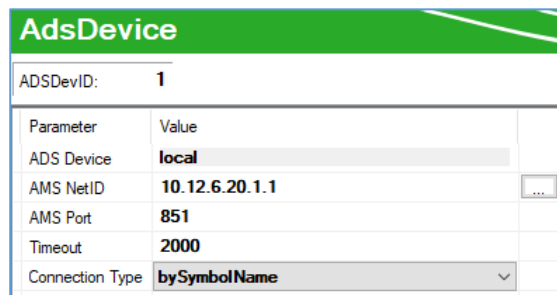
- **Log Mode** – dane mogą być logowane w momencie zmiany wartości zmiennej (*onChange*) lub co określony w milisekundach w parametrze *Cyclic Time* czasu (opcja *Cyclic*)

AutoLogGroup	
AutoLogGrpID:	<b>1</b>
Parameter	Value
StartUp	<b>Manual</b>
Direction	<b>DeviceAsSource</b>
Write Mode	<b>APPEND</b>
Ringbuffer Parameter	0
Log Mode	<b>cyclic</b>
Cycle Time	<b>500</b>

W opisywanym przykładzie konfiguracja wygląda jak na zdjęciu powyżej (Start Up → Manual, Write Mode → Append).

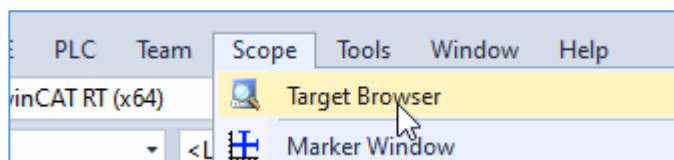
### 2.4.1 AdsDevice

W oknie AdsDevice, w polach **AMS NetID** i **ADS Device**, należy wybrać urządzenie, z którego pobierane będą zmienne. Zmieniając typ połączenia można zdecydować, czy zmienne mają być odszukiwane po adresie w pamięci (**Connection Type** → **byGroupOffset**) czy po nazwie (**Connection Type** → **bySymbolName**). Ma to znaczenie w przypadku ręcznego dodawania symbolów zmiennych. Przy odszukiwaniu po adresie w trakcie deklarowania kolejnych zmiennych adresy mogą się zmieniać, co może potem powodować nieprzewidziane skutki. Przy odszukiwaniu po nazwie ten problem nie występuje.

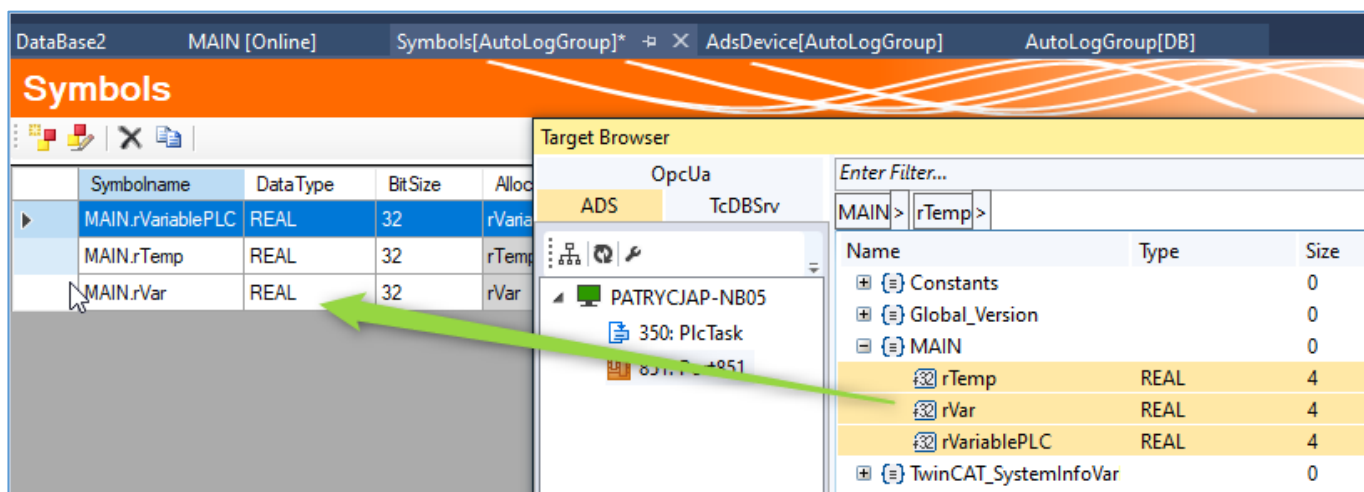


### 2.4.2 Symbols

Zakładka **Symbols** służy do skonfigurowania zmiennych, które mają być logowane do bazy. Można w tym celu skorzystać z narzędzia **Target Browser**. Program na sterowniku, z którego będą pobierane zmienne musi być uruchomiony.



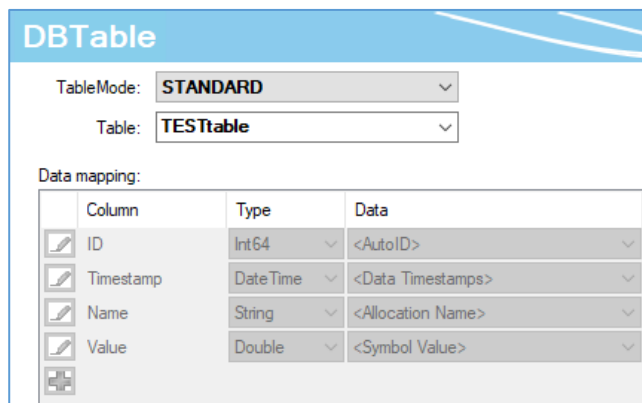
Wybrane zmienne należy z okna **Target Browser** umieścić w zakładce **Symbols** metodą drag'n'drop.



### 2.4.3 DBTable

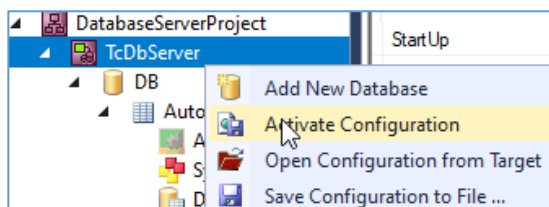
Ostatnim krokiem jest dodanie standardowej tabeli w zakładce **DBTable**. Do tej tabeli wpisywane będą zmienne. W wersji **STANDARD** każda zmienna zostanie przekazana osobno. Można również wybrać tryb **CUSTOM**, który pozwoli na ręczne dopasowanie nazwy, typu i zawartości każdej kolumny, jednak trzeba pamiętać, aby były one zgodne z istniejącą tabelą.

W przykładzie wybrano **TableMode** → **Standard** i **Table** → (Nazwa utworzonej wcześniej tabeli).



### 2.4.4 Uruchomienie

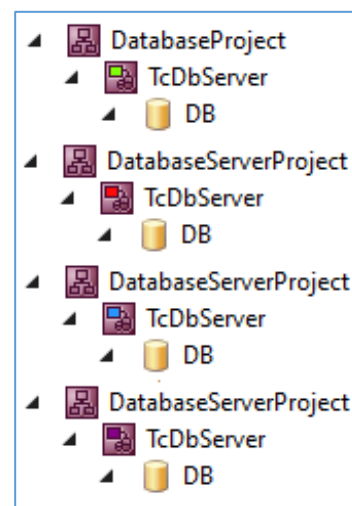
**Uwaga!** Po zakończeniu konfiguracji należy zapisać wszystkie zmiany wybierając **Save All**, niezapisane zmiany nie zostaną wprowadzone. Następnie należy aktywować konfigurację projektu bazy danych.



Status serwera można sprawdzić obserwując kolor ikony w drzewie projektu.

Dozwolone kolory to:

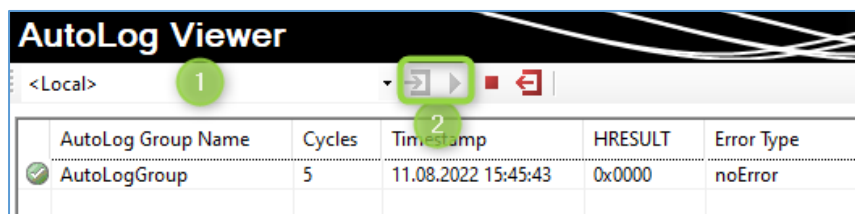
- Zielony – gotowy do pracy
- Czerwony – nie można połączyć z TC3 DataBase Server
- Niebieski – brak licencji Tc3\_Database
- Fioletowy – Grupa autologowania jest w trakcie pracy



Jeżeli aktywacja konfiguracji przebiegła pomyślnie, można uruchomić grupę autologowania. Należy w tym celu otworzyć okno **AutoLog Viewer** zieloną ikoną z Toolbara dodatku Database Server:



W opisywanym przykładzie zarządzanie grupą autologowania zostało ustawione na tryb Manualny. Pozwala to na ręcznie uruchamianie/zatrzymywanie grupy. W oknie **AutoLog Viewer** najpierw należy wybrać Serwer (1) a następnie uruchomić logowanie (2). Ewentualne błędy pojawią się w kolumnach HRESULT i Error Type.

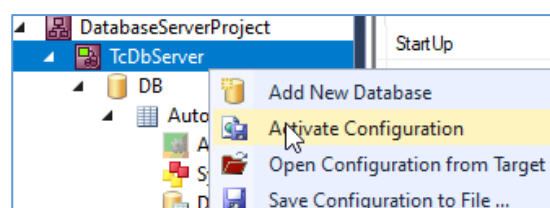
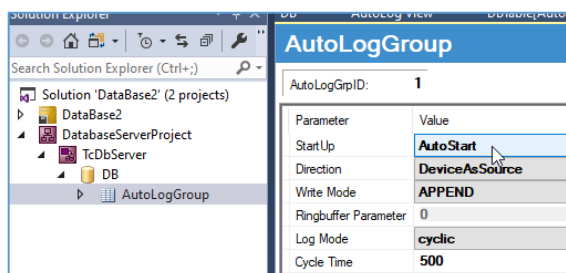


Jeśli kilka cykli zaloguje się poprawnie, grupę można zatrzymać. Aby odczytać dane wpisane do bazy, należy wykonać komendę **Select**, jak opisano to we wcześniejszej części instrukcji (w oknie **Sql Query Editor**).

Przykład odczytanych danych:

Result	
Name	Value
Dataset[0]	
Dataset[1]	
Dataset[2]	
Dataset[3]	
ID	4
Timestamp	19.07.2022 15:12:13
Name	rTemp
Value	0
Dataset[4]	
Dataset[5]	
Dataset[6]	
Dataset[7]	
ID	8
Timestamp	19.07.2022 15:12:14
Name	rVar
1 to (32)	
Batch 10	

Jeśli grupa autologowania działa poprawnie, można zmienić tryb jej uruchamiania na **AutoStart** i aktywować konfigurację projektu baz danych. Od tej pory skonfigurowane zmienne będą logować się do bazy automatycznie, razem z uruchomieniem się Run-Time'u TwinCATa.



### 3 Tryb konfiguracji – przykład niestandardowej tabeli

W trybie konfiguracyjnym można skorzystać z opcji stworzenia własnej tabeli. W tym celu należy postępować jak w rozdziale 2.3, ale na etapie tworzenia tabeli, zamiast opcji *Set Standard Table Schema*, należy dodać odpowiednie pola opcją **Append Datafield**. W każdej tabeli niezbędny jest klucz główny, dlatego zaleca się dodanie ID w pierwszej kolumnie. Wybranie typu *BigInt* oraz własności *IDENTITY(1,1)* spowoduje, że ID będzie miało wartość początkową równą 1 i będzie inkrementowane automatycznie przez serwer. Stempel czasowy można uzyskać poprzez dodanie kolumny *Timestamp* typu *DateTime*. Analogicznie należy dodać kolejne kolumny.

Name	Create query	Length	String Length	Properties
ID	BigInt	8	0	IDENTITY(1,1)
Timestamp	DateTime	4	0	
Val_bit	Bit	1	0	
Val_int	Integer	4	0	
Val_bigint	BigInt	8	0	
Val_real	Real	4	0	
Val_nvarchar50	NVarChar	50	49	

```
--AUTOGENERATED SQL CREATE TABLE STATEMENT FOR CustomTable

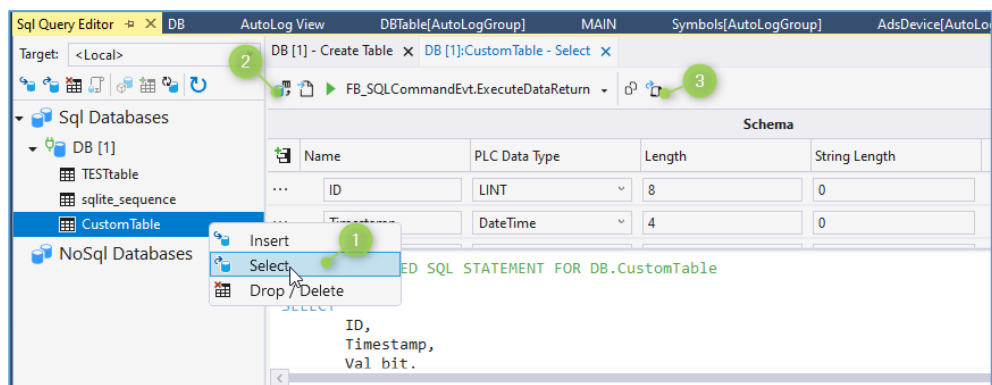
CREATE TABLE "CustomTable"
(
    "ID" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    "Timestamp" DATETIME,
    "Val_bit" BOOLEAN,
    "Val_int" INT,
    "Val_bigint" INTEGER,
    "Val_real" REAL,
    "Val_nvarchar50" VARCHAR(49)
)
```

#### 3.1 Rzutowanie zmiennych

Okno **SQL Query Editor** pozwala na eksport utworzonej tabeli do struktury zmiennych dla programu PLC. Dzięki temu zyskujemy pewność, że przygotowane do logowania w programie PLC zmienne będą odpowiedniego typu. Aby wykonać eksport należy, po utworzeniu tabeli, należy:

- kliknąć na tabelę PPM i wybrać opcję **Select** (1)
- wykonać komendę **Get Table Schema** (2)
- wykonać eksport opcją **Export Tablestruct to TwinCAT3 DUT** (3)





Plik można zapisać w dowolnym miejscu na dysku komputera.

Aby zaimportować strukturę do projektu PLC należy kliknąć **PPM** na folder **DUT** i wybrać opcję **Import PLCOpenXML...** Po zaimportowaniu wcześniej wyeksportowanej struktury widać, że typy zmiennych SQL zostały zamienione na odpowiadające im typy zmiennych PLC.

Następnie należy strukturę takiego typu zadeklarować w programie PLC oraz wgrać i uruchomić program.

```

1  TYPE ST_SelectStruct :
2  STRUCT
3      ID: LINT;
4      Timestamp: DT;
5      Val_bit: BOOL;
6      Val_int: DINT;
7      Val_bigint: LINT;
8      Val_real: REAL;
9      Val_nvarchar50: STRING(50);
10 END_STRUCT
11 END_TYPE

```

### 3.2 Niestandardowa grupa autologowania

Aby uruchomić niestandardową grupę autologowania należy skonfigurować zakładki **AutoLogGroup** oraz **AdsDevice** jak opisano w rozdziale 2.4. W zakładce **Symbols**, korzystając z okna **Target Browser**, należy umieścić zawartość opisanej wyżej struktury z programu PLC (program PLC musi być uruchomiony):

Symbols						
	Symbolname	Data Type	Bit Size	AllocationName	IndexGroup	IndexOffset
	MAIN.st_dbtest.ID	LINT	64	st_dbtest.ID	16448	449648
	MAIN.st_dbtest.Timestamp	DT	32	st_dbtest.Timestamp	16448	449656
	MAIN.st_dbtest.Val_bit	BOOL	8	st_dbtest.Val_bit	16448	449660
	MAIN.st_dbtest.Val_int	DINT	32	st_dbtest.Val_int	16448	449664
	MAIN.st_dbtest.Val_bigint	LINT	64	st_dbtest.Val_bigint	16448	449672
	MAIN.st_dbtest.Val_real	REAL	32	st_dbtest.Val_real	16448	449680
	MAIN.st_dbtest.Val_nvarchar50	STRING	400	st_dbtest.Val_nvarchar50	16448	449684

W zakładce **DBTable** w polu **TableMode** należy wybrać opcję **CUSTOM**, a w polu **Tables** wybrać odpowiednią tabelę. Następnie trzeba ręcznie wykonać powiązanie zmiennych:

DBTable

TableMode: CUSTOM

Table: Table

Data mapping:

	Column	Type	Data
	ID	Int64	<AutoID>
	Timestamp	DateTime	<Data Timestamps>
	Val_bit	Boolean	MAIN.st_dbtest.Val_bit
	Val_int	Int32	MAIN.st_dbtest.Val_int
	Val_bigint	Int64	MAIN.st_dbtest.Val_bigint
	Val_real	Single	MAIN.st_dbtest.Val_real
	Val_nvarchar50	String	MAIN.st_dbtest.Val_nvarchar50

Jeżeli grupa ma za zadanie zapisywać wartości do bazy to zmienne *ID* oraz *Timestamp* należy odpowiednio ustawić jako *<AutoID>* oraz *<Data Timestamps>*. Jeżeli jednak grupa będzie miała za zadanie odczytywanie zmiennych z bazy to *<AutoID>* oraz *<Data Timestamps>* należy zastąpić zmiennymi ze struktury, do których wpisywane będą odczytywane wartości.

Na koniec należy wykonać zapis (Save All) i aktywację konfiguracji projektu baz danych.

## 4 Sterowanie grupami autologowania poprzez PLC

W programie PLC możemy w dowolnym momencie sprawdzić status lub rozpocząć/przerwać pracę każdej grupy autologowania. Służy do tego blok *FB\_PLCDBAutoLogEvt* oraz metody *RunOnce()*, *Start()*, *Status()*, *Stop()*. Aby rozróżnić konkretne bazy oraz grupy potrzebujemy *DBID* otrzymane przy konfiguracji serwera bazy oraz *AutoLogGrpID* otrzymane podczas dodawania grupy.

[HTTPS://INFOSYS.BECKHOFF.COM/ENGLISH.PHP?CONTENT=../CONTENT/1033/TF6420\\_TC3\\_DATABASE\\_SERVER/2674373259.HTML](https://infosys.beckhoff.com/english.php?content=../content/1033/TF6420_TC3_DATABASE_SERVER/2674373259.html)

## 5 Tryb PLC Expert

Tryb ten ma podobne możliwości oraz wymagania, co tryb konfiguracji. Jedyną różnicą jest wprowadzanie wszystkich wyżej wymienionych ustawień bezpośrednio w kodzie PLC. Wszystkie bloki funkcyjne zawierają metody pozwalające na wykonanie wielu różnych komend SQL w zależności od zapotrzebowania. Skrócone opisy możliwych do zrealizowania czynności:

- *FB\_ConfigTcDBSrv* – Wprowadza zmiany w pliku konfiguracyjnym *CurrentConfigDataBase.xml*. Plik ten zawiera informacje o całej konfiguracji *Tc3\_DataBase\_Server* na danym komputerze i znajduje się domyślnie w folderze *C:\TwinCAT\3.1\Boot*. Za pomocą tego bloku możemy więc dowolnie odczytywać, dodawać oraz usuwać kolejne bazy danych oraz grupy autologowania zmiennych.

- *FB\_PLCDBAutoLogEvt* – Blok ten pozwala na uruchamianie, zatrzymywanie oraz sprawdzanie statusu grup autologowania.

- *FB\_PLCDBCCreateEvt* – Pozwala na utworzenie pliku bazy danych określonego wcześniej w pliku konfiguracyjnym, oraz nowych tabeli do dowolnych baz danych.

- *FB\_PLCDBReadEvt* – Działa w dwóch trybach. Pierwszy pozwala na odczytywanie dowolnej ilości rejestrów z tabeli w formacie zgodnym ze standardową tabelą *Tc3\_Database*. Drugi tryb pozwala na odczytanie dowolnej tabeli, ale wymaga załączenia struktury zgodnej z kolejnymi kolumnami tabeli.

- *FB\_PLCDBWriteEvt* – Pozwala na zapis wartości w trzech trybach. Pierwsze dwa zapisują wartości zmiennych do tabeli zgodnej ze standardową tabelą *Tc3\_Database*, odpowiednio po nazwie zmiennej oraz po adresie ADS. Trzeci zapisuje w tabeli zawartość dowolnej zgodnej z nią struktury.

- *FB\_PLCDBCmdEvt* – Pozwala na wykonywanie bardziej zaawansowanych funkcji odczytu oraz zapisu. Możliwe jest między innymi odczytywanie/zapisywanie tylko części kolumn z danego wpisu oraz selektywne odczytywanie rekordów przy pomocy komendy *WHERE*. Blok ten wysyła do serwera bazy danych bezpośrednią komendę SQL, którą możemy ręcznie zdefiniować. W komendzie tej mogą znajdować się placeholdery, które następnie ustawiamy na wejście bloku w formie tablicy struktur *ST\_ExpParameter*. Dokładne możliwości tego bloku zależne są od obsługiwanych przez daną bazę komendami SQL.

Bloki te posiadają na wyjściu interfejs *ipTcResultEvent* zgodny z formułą *EventLoggera*. Pozwala to na bardzo łatwe przetwarzanie wiadomości o błędach. Przy pomocy biblioteki *Tc3\_EventLogger* wiadomości z bloków można wpisać do zmiennych o specjalnym typie. Skorzystamy z tego w następnych przykładach.

Uruchamiamy program **P\_PLCExpert\_AutoLog**. Jest to przykład wykorzystania trybu PLC Expert zrobiony według schematu obok.

Zmiana zmiennej **bRun** na TRUE spowoduje uruchomienie grupy autologowania.

bRun	BOOL	FALSE
bStop	BOOL	FALSE
tAutoLogStatusCheckCycle	TIME	T#3s

W *Sql Query Editor* można zaobserwować pojawiające się rekordy.

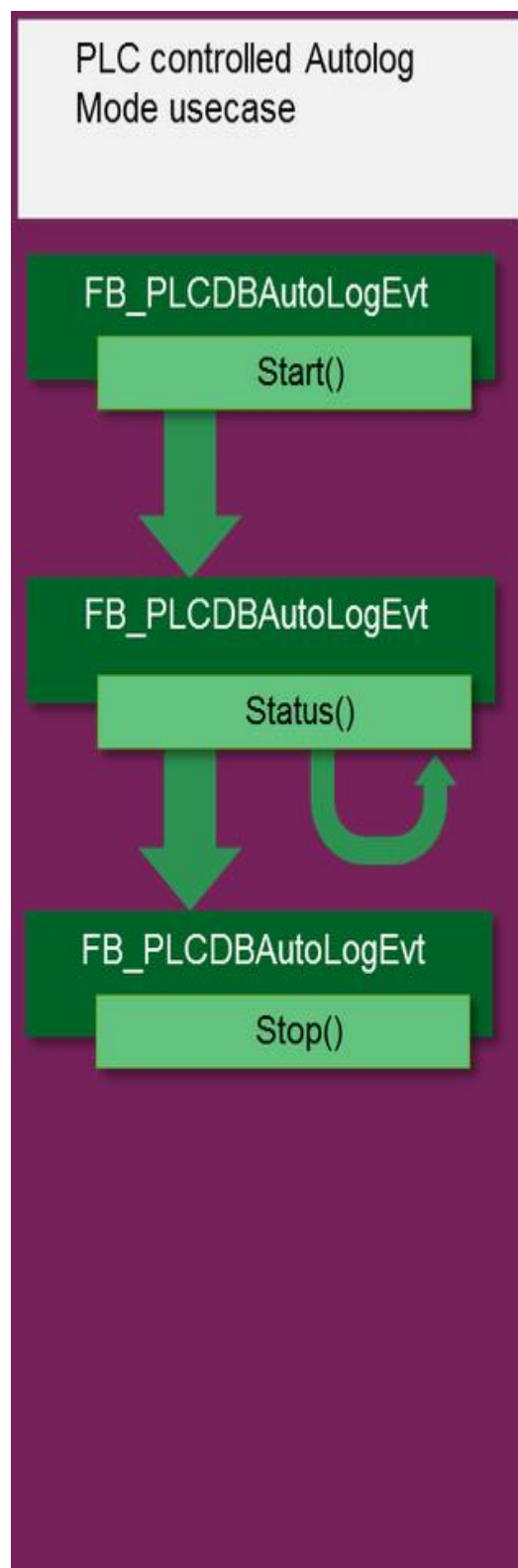
Result	
Name	Value
▶ Dataset[884]	
▶ Dataset[885]	
▶ Dataset[886]	
▶ Dataset[887]	
▶ Dataset[888]	
<div> <div>89</div> <div>to (89)</div> <div>Batch 10</div> </div>	

Kiedy zbierzemy zadowalającą nas liczbę danych, możemy zmienić **bStop** na TRUE, żeby zatrzymać grupę autologowania.

Możemy też zmienić okres co jaki aktualizowany jest status grupy za pomocą zmiennej **tAutoLogStatusCheckCycle**.

Status można podejrzeć w zmiennej **ipTcResult**.

ipTcResult	Tc3_Database.I_Tc...	16#FFF...
eSeverity	TCEVENTSEVERITY	Verbose
ipSourceInfo	I_TcSourceInfo	16#FFF...
nEventId	UDINT	0
sEventClassName	STRING(255)	'Success'
sEventText	STRING(255)	'Success'



Włączamy drugi przykład **P\_PLCExpert\_Independent\_FBs**, którego schemat wykonania widać obok.

Obsługa programu polega na modyfikowaniu podanych zmiennych.

bRun	BOOL	FALSE
hDBID	UDINT	1
sDBID	STRING(50)	'TESTta...
hAutoLogGroup	UDINT	1
nRecords	UDINT	10
nInitialIndex	UDINT	0
nRecordAmount	UDINT	20
sOrderBy	STRING(50)	'ID'
eCommands	E_COMMANDS	ReadAlif...
ePLCDBState	E_PLCDBSTATE	Wait
stDB	ST_DB	

Zmienne **hDBID** i **sDBID** odpowiadają za określenie, na której bazie danych i tabeli będą wykonywane operacje. **Ustawiamy odpowiednio na 1 i TESTtable**. W razie tworzenia nowej tabeli będą to jej parametry, dlatego będzie trzeba zmienić nazwę.

Operacje uruchamiamy zmieniając **ePLCDBState**. Dostępne opcje:

**Wait** - żadna operacja nie jest wykonywana,

**WriteData** – wpisanie danych do tabeli i ich odczytanie,

**ReadData** – odczytanie danych z tabeli,

**RunCMDCommand** – wykonanie komendy SQL,

**AutoLog** – jednokrotne uruchomienie grupy autologowania,

**CreateTable** – stworzenie nowej tabeli,

**EventHandling** – wykonywana automatycznie w razie błędu. Wybranie jej ręcznie umożliwia zaktualizowanie i podejrzenie informacji z Event Loggera. Informacje zapisywane są w poniższych zmiennych.

sSourceInfo	Tc3_EventLogger.I_TcSourceInfo
nId	UDINT
sName	STRING((ParameterList.cSourceNam...
sEventClass	Tc3_EventLogger.GUID
eSeverity	TCEVENTSEVERITY
stEventEntry	Tc3_EventLogger.TcEventEntry
uuidEventClass	GUID
nEventId	UDINT
eSeverity	TCEVENTSEVERITY

## Independent functionsblocks of the PLC Expert Mode

### FB\_PLCDBReadEvt

Read()

ReadStruct()

### FB\_PLCDBWriteEvt

Write()

WriteBySymbol()

WriteStruct()

### FB\_PLCDBCmdEvt

Execute()

ExecuteDataReturn()

### FB\_PLCDBCCreateEvt

Database()

Table()

W *RunCMDCommand* wykonywaną komendę wybiera się za pomocą **eCommands**. Możliwe opcje:

**InsertIntoTable** – wpisuje dane do tabeli,

**ClearTable** – usuwa dane z tabeli,

**ReadAllfromTable** – czyta wszystkie dane z tabeli,

**CustomReadfromTable** – czyta rekordy, w których Value równe jest 21.3,

**DropTable** – usuwa tabelę

Reszta zmiennych to parametry, z których korzystają bloki funkcyjne wywoływane za pomocą *ePLCDBState*. Ich nastawy przedstawione zostały poniżej.

Wszelkie dane pobierane z tabeli są wpisywane do tablicy **arrData**.

hAutoLogGroup	UDINT	1
nRecords	UDINT	10
nInitialIndex	UDINT	0
nRecordAmount	UDINT	20
sOrderBy	STRING(50)	'ID'
stDB	ST_DB	
ID	LINT	0
Timestamp	DATE_AND_TIME	DT#1970-1-1-0:0:0
Name	STRING	'Temperature'
Value	LREAL	21.3
arrData	ARRAY [0..19] OF S...	
arrData[0]	ST_DB	
ID	LINT	1
Timestamp	DATE_AND_TIME	DT#2022-7-19-15:12:13
Name	STRING	'rTemp'
Value	LREAL	0
arrData[1]	ST_DB	
arrData[2]	ST_DB	

Możemy przetestować wszystkie operacje i komendy SQL, z wyjątkiem *CreateTable*. W celu wykonania *CreateTable* należy zmienić *sDBID* na dowolną inną nazwę. **Zmieniamy na TESTtableNew**. Efekty wykonanych operacji podejrzeć można w Sql Query Editor i w *arrData*.

## 6 Tryb SQL Expert

Tryb *SQL Expert* jest najbardziej efektywnym trybem. Zezwala on użytkownikowi na wykorzystanie zaawansowanych możliwości, na które zezwala dany typ bazy danych np. procedur przechowywanych w bazie.




Tryb ten można podzielić na cztery zasadnicze działania:

### 6.1 Wysłanie komendy SQL bez odczytu danych (INSERT / UPDATE)

Uruchamiamy program *P\_SQLExpert\_NoDataRet*. Korzysta on z poniższych bloków funkcyjnych:

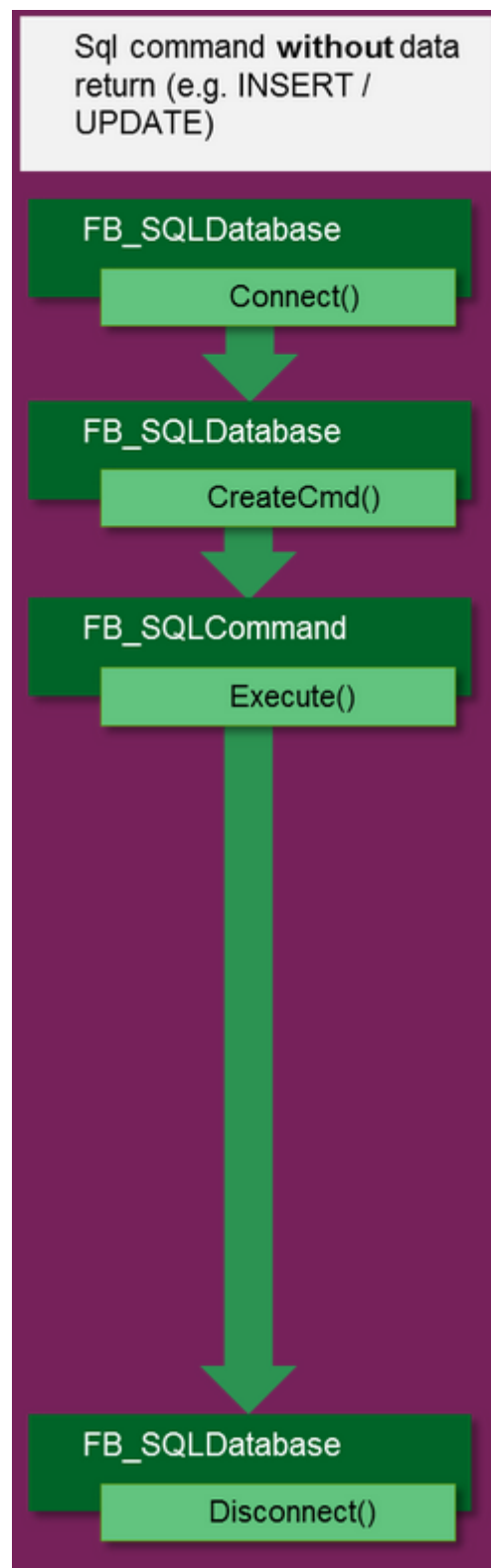
- > *FB\_SQLDatabase.Connect()* – połączenie z bazą danych,
- > *FB\_SQLDatabase.CreateCmd()* – stworzenie komendy SQL,
- > *FB\_SQLCommand.Execute()* – wykonanie komendy,
- > *FB\_SQLDatabase.Disconnect()* – rozłączenie z bazą danych.

Program obsługuje się za pomocą poniższych zmiennych.

 <i>sTableName</i>	STRING	'TESTtable'
 <i>hDBID</i>	UDINT	1
 <i>bInsert</i>	BOOL	FALSE

Zmienna *hDBID* określa numer bazy danych, na której wykonamy przykład. **Ustawiamy na 1.**

*sTableName* określa nazwę tabeli, na której wykonamy przykład, jeśli tabela o takiej nazwie nie istnieje to zostanie utworzona. **Ustawiamy na TESTtable1.**





Ustawienie **blinsert** na TRUE powoduje dodanie tabeli i rekordu takich jak poniżej.

Sql Query Editor

Target: <Local>

DB [1]:TESTtable - Select x DB [1]:TESTtable1 - Select x

FB\_SQLCommandEvt.ExecuteDataReturn

Schema

Name	PLC Data Type	Length	String Length
ID	LINT	8	0
Timestamp	DateTime	4	0
Name	STRING	81	80
Value	LREAL	8	0

```

SELECT
    ID,
    Timestamp,
    Name,
    Value
FROM
    TESTtable1
    
```

Result

Name	Value
Dataset[0]	
ID	1
Timestamp	31.01.2018 14:59:27
Name	Temperature
Value	21,3

1 to (1) Batch 10

## 6.2 Wysłanie komendy SQL z odczytem danych (SELECT)

Uruchamiamy program **P\_SQLExpert\_NoDataRet**. Realizuje on wykonanie komendy SQL - SELECT na określonej tabeli. Korzysta on z poniższych bloków funkcyjnych:

```
-> FB_SQLDatabase.Connect(),
-> FB_SQLDatabase.CreateCmd(),
-> FB_SQLCommand.ExecuteDataReturn() – realizacja komendy i
przygotowanie danych,
-> FB_SQLResult.Read() - odczytanie danych,
-> FB_SQLResult.Release() – zwolnienie danych,
-> FB_SQLDatabase.Disconnect().
```

Obsługa programu zachodzi za pomocą modyfikacji poniższych zmiennych.

hDBID	UDINT	1
bRead	BOOL	FALSE
udiStartIndex	UDINT	0
udiRecordCount	UDINT	1
sTableName	STRING	'TESTtable1'

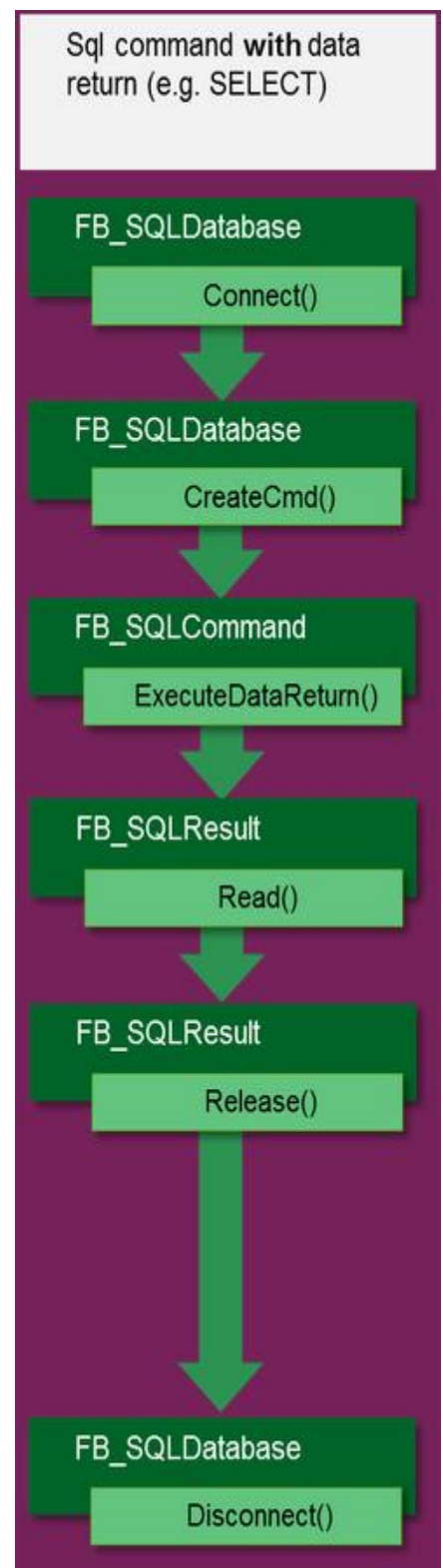
Zmienne **hDBID** i **sTableName** pełnią analogiczną funkcję jak w poprzednim przykładzie. Ustawiamy odpowiednio na **1** i **TESTtable1**.

**udiStartIndex** i **udiRecordCount** określają początkowy indeks, od którego czytane będą dane i liczbę przeczytanych rekordów. Chcemy przeczytać rekord dodany w poprzednim przykładzie więc ustawiamy **udiStartIndex= 1** i **udiRecordCount= 0**.

Ustawiamy **bRead** na **TRUE**, żeby wystartować program.

Wynik komendy SELECT zostanie zawarty w strukturze **stSelect**.

stSelect	ST_Select	
ID	LINT	1
Timestamp	DATE_AND_TIME	DT#2018-1-31-14:59:27
Name	STRING(80)	'Temperature'
Value	LREAL	21.3



## Użycie procedury bez odczytu danych.

```
-> FB_SQLDatabase.Connect()  
-> FB_SQLDatabase.CreateSP()*  
-> FB_SQLStoredProcedure.Execute()  
-> FB_SQLStoredProcedure.Release()  
-> FB_SQLDatabase.Disconnect()
```

## Użycie procedury z odczytem danych.

```
-> FB_SQLDatabase.Connect()  
-> FB_SQLDatabase.CreateSP()*  
-> FB_SQLStoredProcedure.ExecuteDataReturn()  
-> FB_SQLResult.Read()  
-> FB_SQLResult.Release()  
-> FB_SQLStoredProcedure.Release()  
-> FB_SQLDatabase.Disconnect()
```

\*procedurę tworzy się tylko raz, przesyłane zostają jedynie jej parametry.