**Работа с SQL. Шаблон общего модуля**

Публикация № **1537962**

**Шаблон общего модуля для работы с MS/PG SQL из 1С.**

В одной из последних задач появилась надобность в запросах к SQL базам из 1С. Для этого был написан общий модуль, код из которого предлагаю ниже.

Думаю, для кого то это может быть не плохим подспорьем в выполнении схожих задач. Критика ~~и моральные унижения~~ приветствуются.

**Основные функции:**

*Запрос в СУБД*

**СоздатьДанныеДляПодключения**- конструктор данных для подключения к бд. Сюда передаются логин, пароль и т.д. Используется в НовоеСоединениеССУБД.

**НовоеСоединениеССУБД**- создает соединение с бд COMОбъект("ADODB.Connection").

**ВыполнитьЗапрос**- выполняет запрос и возвращает результат типа COMОбъект.

**ПолучитьРезультатЗапроса**- выполняет запрос и возвращает результат типа Таблица значений.

**ВыполнитьЗапросВ\_Personnel**- пример запроса к бд.

**ЗакрытьСоединениеССУБД**- закрывает соединение с бд.

**СоздатьПараметрЗапроса**- создает параметры для передачи в запрос. Используется в ВыполнитьЗапрос и ПолучитьРезультатЗапроса, передается в данные функции в виде результата выполнения СоздатьПараметрЗапроса, либо, для передачи нескольких параметров, массив с результатами выполнения СоздатьПараметрЗапроса. По умолчанию тип 12, что позволяет MS SQL самой попытаться подобрать тип передаваемого параметра. В самом запросе параметры указываются как "?", подставляются попорядку добавления.

**ВСтрокуBase64**- Используется для типа данных "adLongVarBinary 205 Long binary value", на пример, для получения данных типа image sql в формате Base64. В примере ниже используется для получения изображения из бд.

*Конструкторы запроса*

**ЗапросДелет**- конструктор запроса для удаления данных из бд. Для страховки, по умолчанию условия установлено в ложь. Пример использования есть ниже.

**ЗапросИнсерт**- конструктор запроса для добавления данных из бд. Пример использования есть ниже.

**ЗапросДобавитьИлиОбновить\_My\_MS\_SQL**/ **ЗапросДобавитьИлиОбновить\_Pg\_SQL**- конструктор запроса для добавления, или обновления данных данных из бд. Пример использования есть ниже.

**ФорматЧисла**/ **ФорматБулевоPG**/ **ФорматДатыВремяPG** / **ФорматДатыPG**/ **ФорматВремяPG**/ **ВСтроку**- функции форматирования данных для SQL запросов.

**Код модуля**

////////////////////////////////////////////////////////////////////////////////

// Серверные процедуры и функции для работы с SQL запросами:

////////////////////////////////////////////////////////////////////////////////

#Область ПрограммныйИнтерфейс

#Область Конструктор\_запросов\_в\_SQL

// Возвращает строку запроса для удаления набора значений в таблице SQL.

// Параметры:

// Таблица - строка - имя таблицы.

// Условия - строка - условия отбора в части "WHERE".

//

// Возвращаемое значение:

// Строка - строка запроса

Функция ЗапросДелет(Таблица, Условия = "false") Экспорт

Запрос = Новый Массив;

Запрос.Добавить("DELETE FROM ");

Запрос.Добавить(Таблица);

Запрос.Добавить(" WHERE ");

Запрос.Добавить(Условия);

Возврат СтрСоединить(Запрос);

КонецФункции

// Возвращает строку запроса для добавления набора значений в таблицу SQL.

// Параметры:

// Таблица - строка - имя таблицы.

// Поля - строка - имена полей через запятую.

// Строки - массив строк - в каждом элементе массива находится строка со значениями полей по порядку, указанному в параметре "Поля".

// Пример: Строки = Новый Массив;

// Строки.Добавить("'DDD', 30, 30, 2, '20210201'");

// Строки.Добавить("'YYY', 30, 30, 2, '20210201'");

//

// Возвращаемое значение:

// Строка - строка запроса

Функция ЗапросИнсерт(Таблица, Поля, Строки) Экспорт

Запрос = Новый Массив;

Запрос.Добавить("INSERT INTO");

Запрос.Добавить(Таблица);

Запрос.Добавить("(");

Запрос.Добавить(Поля);

Запрос.Добавить(")");

Запрос.Добавить("VALUES");

Запрос.Добавить(СобратьДобавляемыеСтроки(Строки));

Возврат СтрСоединить(Запрос, " ");

КонецФункции

// Возвращает строку для запроса с отформатированным числом.

//

// Параметры:

// Число - число - число, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированным числом

Функция ФорматЧисла(Число) Экспорт

Возврат Формат(Число, "ЧН=0; ЧГ=");

КонецФункции

// Возвращает строку запроса для добавления набора значений, или обновлении при наличие в таблицу SQL.

// Параметры:

// Таблица - строка - имя таблицы.

// Поля - строка - имена полей через запятую.

// Строки - массив строк - в каждом элементе массива находится строка со значениями полей по порядку, указанному в параметре "Поля".

// Пример: Строки = Новый Массив;

// Строки.Добавить("'DDD', 30, 30, 2, '20210201'");

// Строки.Добавить("'YYY', 30, 30, 2, '20210201'");

// ОбновляемыеПоля - строка - поля для обновление в случае, если найдена запись по ключу

//Пример: "Колонка1 = VALUES(Колонка1), Колонка2 = VALUES(Колонка2)"

//"Колонка1 = VALUES(Колонка1) + VALUES(Колонка2)"

//

// Возвращаемое значение:

// Строка - строка запроса

Функция ЗапросДобавитьИлиОбновить\_My\_MS\_SQL(Таблица, Поля, Строки, ОбновляемыеПоля) Экспорт

//"INSERT INTO `table` (`id`,`a`,`b`,`c`) VALUES (?,?,?)

ON DUPLICATE KEY UPDATE `a` = ?, `b` = ? `c` = ?";

Запрос = Новый Массив;

Запрос.Добавить("INSERT INTO");

Запрос.Добавить(Таблица);

Запрос.Добавить("(");

Запрос.Добавить(Поля);

Запрос.Добавить(")");

Запрос.Добавить("VALUES");

Запрос.Добавить(СобратьДобавляемыеСтроки(Строки));

Запрос.Добавить("ON DUPLICATE KEY UPDATE");

Запрос.Добавить(ОбновляемыеПоля);

Возврат СтрСоединить(Запрос, " ");

КонецФункции

// Возвращает строку запроса для добавления набора значений, или обновлении при наличие в таблицу SQL.

//

// Параметры:

// Таблица - строка - имя таблицы.

// Поля - строка - имена полей через запятую.

// Строки - массив строк - в каждом элементе массива находится строка со значениями полей по порядку, указанному в параметре "Поля".

// Пример: Строки = Новый Массив;

// Строки.Добавить("'DDD', 30, 30, 2, '20210201'");

// Строки.Добавить("'YYY', 30, 30, 2, '20210201'");

// ОбновляемыеПоля - строка - поля для обновление в случае, если найдена запись по ключу

//Пример: "Колонка1 = VALUES(Колонка1), Колонка2 = VALUES(Колонка2)"

//"Колонка1 = VALUES(Колонка1) + VALUES(Колонка2)"

// Ключи - строка - ключевые поля через запятую по которым идет поиск для обновления

//

// Возвращаемое значение:

// Строка - строка запроса

Функция ЗапросДобавитьИлиОбновить\_Pg\_SQL(Таблица, Поля, Строки, ОбновляемыеПоля, Ключи) Экспорт

Запрос = Новый Массив;

Запрос.Добавить("INSERT INTO");

Запрос.Добавить(Таблица);

Запрос.Добавить("(");

Запрос.Добавить(Поля);

Запрос.Добавить(")");

Запрос.Добавить("VALUES");

Запрос.Добавить(СобратьДобавляемыеСтроки(Строки));

Запрос.Добавить("ON CONFLICT (");

Запрос.Добавить(Ключи);

Запрос.Добавить(") DO UPDATE SET ");

Запрос.Добавить(ОбновляемыеПоля);

Возврат СтрСоединить(Запрос, " ");

КонецФункции

// Возвращает строку для запроса с отформатированным булево.

//

// Параметры:

// Булево - булево - значение булево, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированным булево

Функция ФорматБулевоPG(Булево) Экспорт

Возврат ?(Булево, "true", "false");

КонецФункции

// Возвращает строку для запроса с отформатированным Дата + время.

//

// Параметры:

// Дата - дата - значение Дата, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированным Дата + время

Функция ФорматДатыВремяPG(Дата) Экспорт

ФорматДата = Формат(Дата, "ДФ=yyyyMMddHHmm; ДП=NULL");

Возврат ?(ФорматДата = "NULL", ФорматДата, ВСтроку(ФорматДата));

КонецФункции

// Возвращает строку для запроса с отформатированной Датой.

//

// Параметры:

// Дата - дата - значение Дата, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированной Датой

Функция ФорматДатыPG(Дата) Экспорт

ФорматДата = Формат(Дата, "ДФ=yyyyMMdd; ДП=NULL");

Возврат ?(ФорматДата = "NULL", ФорматДата, ВСтроку(ФорматДата));

КонецФункции

// Возвращает строку для запроса с отформатированным Временем.

// Параметры:

// Дата - дата - значение Дата, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированным Временем

Функция ФорматВремяPG(Дата) Экспорт

ФорматДата = Формат(Дата, "ДФ=HHmmss; ДП=NULL");

Возврат ?(ФорматДата = "NULL", ФорматДата, ВСтроку(ФорматДата));

КонецФункции

// Возвращает строку для запроса в формате строка.

// Параметры:

// Данные - произвольный - произвольное значение, которое надо отформатировать.

//

// Возвращаемое значение:

// Строка - строка с отформатированным значением

//

Функция ВСтроку(Данные) Экспорт

Возврат "'" + Данные + "'";

КонецФункции

#КонецОбласти

#Область Подключение\_к\_SQL

#Область Готовые\_решения

// Выполнение текста запроса к базе Personnel и возвращает полученные данные

// Параметры:

// ТекстЗапроса - строка - текст запроса.

// Параметры - структура - см. СоздатьПараметрЗапроса().

// Отказ - булево - при ошибке в выполнении функции- истина

// ОписаниеОшибки - строка - описание ошибки выполнения функции

//

// Возвращаемое значение:

// Таблица значений - таблица значений с результатом запроса из базы Personnel

//

Функция ВыполнитьЗапросВ\_Personnel(ТекстЗапроса, Параметры = Неопределено, Отказ = Ложь, ОписаниеОшибки = Неопределено) Экспорт

Connection = НовоеСоединениеССУБД(СоздатьДанныеДляПодключения("172.00.0.00", "Personnel", "log", "pass"), Отказ, ОписаниеОшибки);

ТЗ = ПолучитьРезультатЗапроса(ТекстЗапроса,, Connection, Параметры,, Отказ, ОписаниеОшибки);

ЗакрытьСоединениеССУБД(Connection, Отказ, ОписаниеОшибки);

Возврат ТЗ;

КонецФункции

#КонецОбласти

// Возвращает структуру с данными для НовоеСоединениеССУБД().

// Параметры:

// АдресСервера - строка - адрес сервера.

// ИмяБД - строка - текст имя базы данных.

// Логин - строка - логин.

// Пароль - строка - пароль.

// ТипСУБД - строка - тип субд к которой совершается подключение

// "MSSQL" для microsoft sql

// "PGSQL" для postgresql

// Порт - строка - порт для подключения

// DataSource - строка

//

// Возвращаемое значение:

// Структура - структура с данными для подключения см. НовоеСоединениеССУБД

//

Функция СоздатьДанныеДляПодключения(АдресСервера, ИмяБД, Логин, Пароль, ТипСУБД = "MSSQL", Порт = 5432, DataSource = "") Экспорт

Возврат Новый Структура("АдресСервера, ИмяБД, Логин, Пароль, ТипСУБД, Порт, DataSource",

АдресСервера, ИмяБД, Логин, Пароль, ТипСУБД, Формат(Порт, "ЧГ=0"), DataSource);

КонецФункции

// Возвращает результат выполнения запроса.

// Параметры:

// SQLText - текст - текст запроса.

// ДанныеДляПодключения - структура - формируется в СоздатьДанныеДляПодключения().

// Connection - COMОбъект - ("ADODB.Connection") коннект к бд.

// Параметры - структура - см. СоздатьПараметрЗапроса().

// LockType - число - Тип блокировки

// - 1 - adLockReadOnly - записи в Recordset будут доступны только на чтение, вы не сможете их изменять. Это значение используется по умолчанию.

// - 2 - adLockPessimistic - наиболее надежный с точки зрения целостности данных вид блокировки. Вы можете изменять записи в Recordset,

// но при начале изменения записи она блокируется на источнике таким образом, что другие пользователи не смогут обратиться к ней ни на чтение,

// ни на запись, пока вы не вызовете методы Update или CancelUpdate.

// - 3 - adLockOptimistic - это значение позволяет выиграть в производительности за счет проигрыша в надежности обеспечения целостности данных.

// Запись на источнике блокируется только на время выполнения метода Update. Остальные пользователи могут одновременно с вами читать и изменять данные на источнике.

// - 4 - adLockBatchOptimistic - то же самое, что обычное оптимистичное, но вместо немедленного обновления по одной записи используется пакетное обновление.

// В ситуации, когда изменяется большое число записей, такое решение позволяет выиграть в производительности.

// ОписаниеОшибки - строка - описание ошибки выполнения функции

// Отказ - булево - если передается истина, функция не выполняется, при ошибке в выполнении функции- истина

//

// Возвращаемое значение:

// COMОбъект - COMОбъект c результатом запроса

//

Функция ВыполнитьЗапрос(SQLText, ДанныеДляПодключения = Неопределено, Connection = Неопределено, Параметры = Неопределено, LockType = 3, ОписаниеОшибки = Неопределено, Отказ = Ложь) Экспорт

Попытка

Если Connection = Неопределено Тогда

Connection = НовоеСоединениеССУБД(ДанныеДляПодключения);

КонецЕсли;

Command = Новый COMОбъект("ADODB.Command");

Command.ActiveConnection = Connection;

Command.CommandText = SQLText;

Если Параметры <> Неопределено Тогда

УстановитьПараметрыЗапроса(Command, Параметры);

КонецЕсли;

RecordSet = Новый ComObject("ADODB.RecordSet");

RecordSet.CursorLocation = 3;

RecordSet.LockType = LockType;

Recordset = Command.Execute();

Возврат Recordset;

Исключение

Отказ = Истина;

ОписаниеОшибки = ОписаниеОшибки();

Возврат Неопределено;

КонецПопытки;

КонецФункции

// Возвращает результат запроса в таблице значений.

// Параметры:

// SQLText - текст - текст запроса.

// ДанныеДляПодключения - структура - формируется в СоздатьДанныеДляПодключения() (необязательный).

// Connection - COMОбъект - ("ADODB.Connection"), коннект к бд (необязательный).

// Параметры - структура - см. СоздатьПараметрЗапроса().

// LockType - число - Тип блокировки (подробнее в функции ВыполнитьЗапрос)

// Отказ - булево - если передается истина, функция не выполняется, при ошибке в выполнении функции- истина

// ОписаниеОшибки - строка - описание ошибки выполнения функции

//

// Возвращаемое значение:

// Таблица значений - таблица значений с результатом запроса

//

Функция ПолучитьРезультатЗапроса(SQLText, ДанныеДляПодключения = Неопределено,

Connection = Неопределено, Параметры = Неопределено,

LockType = 3, Отказ = Ложь, ОписаниеОшибки = Неопределено) Экспорт

ТЗ = Неопределено;

Если Не Отказ Тогда

Recordset = ВыполнитьЗапрос(SQLText, ДанныеДляПодключения,

Connection, Параметры, LockType, ОписаниеОшибки, Отказ);

Если Не Отказ Тогда

Попытка

ТЗ = Новый ТаблицаЗначений;

КоличествоПолей = RecordSet.Fields.Count;

Для НомерСтолбца = 0 По КоличествоПолей - 1 Цикл //Создание и добавление колонок во временную таблицу

ИмяСтолбца = RecordSet.Fields

.Item(НомерСтолбца).Name;

ТипКолонки = SQLТипВ1СТип(RecordSet.Fields

.Item(НомерСтолбца).Type,

RecordSet.Fields.Item(НомерСтолбца).DefinedSize);

Если ТипКолонки = Неопределено Тогда

ТЗ.Колонки.Добавить(RecordSet.Fields

.Item(НомерСтолбца).Name);

Иначе

ТЗ.Колонки.Добавить(RecordSet.Fields

.Item(НомерСтолбца).Name, ТипКолонки);

КонецЕсли;

КонецЦикла;

Пока НЕ RecordSet.EOF Цикл //Заполнение созданной таблицы

НоваяСтрока = ТЗ.Добавить();

Для ПолеСч = 0 По КоличествоПолей - 1 Цикл

Поле = RecordSet.Fields.item(ПолеСч);

НоваяСтрока[Поле.name] = Поле.Value;

КонецЦикла;

RecordSet.MoveNext();

КонецЦикла;

RecordSet.Close();

Возврат ТЗ;

Исключение

Отказ = Истина;

ОписаниеОшибки = ОписаниеОшибки();

ТЗ = Неопределено;

КонецПопытки;

КонецЕсли;

Иначе

ТЗ = Неопределено;

КонецЕсли;

Возврат ТЗ;

КонецФункции

// Возвращает структуру с данными параметра запроса для УстановитьПараметрыЗапроса.

//

// Параметры:

// ИмяПараметра - строка - имя парамета

// Тип - число - тип параметра

// Размер - число - размер значения

// Значение - любое - значение параметра

//

// Возвращаемое значение:

// Структура - структура с данными параметра для запроса

//

Функция СоздатьПараметрЗапроса(ИмяПараметра, Тип = 12, Размер = 1, Значение) Экспорт

//Типы

//adArray 0x2000 Combine with another data type to indicate

// that the other data type is an array

//adBigInt 20 8-byte signed integer 64-битное целое

//adBinary 128 Binary Двоичное

//adBoolean 11 True or false Boolean Булево

//adBSTR 8 Null-terminated character string

//adChapter 136 4-byte chapter value for a child recordset

//adChar 129 String String

//adCurrency 6 Currency format Денежный формат

//adDate 7 Number of days since 12/30/1899 Количество дней с момента 12/30/1899

//adDBDate 133 YYYYMMDD date format Универсальный формат даты ГГГГММДД

//adDBFileTime 137 Database file time База данных файлов времени

//adDBTime 134 HHMMSS time format Универсальный формат времени HHMMSS

//adDBTimeStamp 135 YYYYMMDDHHMMSS date/time format Дататайм как есть YYYYMMDDHHMMSS

//adDecimal 14 Number with fixed precision and scale Специальный тип 1.0хЕ-28 - 1.0хЕ28 одним словом decimal

//adDouble 5 Double precision floating-point Двойной точности с плавающей точкой

//adEmpty 0 no value Пусто

//adError 10 32-bit error code 32-битный код ошибки

//adFileTime 64 Number of 100-nanosecond intervals since 1/1/1601

//adGUID 72 Globally Unique identifier

//adIDispatch 9 Currently not supported by ADO В настоящее время не поддерживается ADO

//adInteger 3 4-byte signed integer 32-битное целое

//adIUnknown 13 Currently not supported by ADO В настоящее время не поддерживается ADO

//adLongVarBinary 205 Long binary value

//adLongVarChar 201 Long string value

//adLongVarWChar 203 Long Null-terminates string value

//adNumeric 131 Number with fixed precision and scale Число с фиксированной точности и масштаба

//adPropVariant 138 PROPVARIANT automation

//adSingle 4 Single-precision floating-point value 32-битное знаковое одинарной точности с плавающей точкой

//adSmallInt 2 2-byte signed integer 16-битное целое со знаком

//adTinyInt 16 1-byte signed integer 8-битное целое со знаком

//adUnsignedBigInt 21 8-byte unsigned integer 64-битное целое беззнаковое

//adUnsignedInt 19 4-byte unsigned integer 32-битное целое беззнаковое

//adUnsignedSmallInt 18 2-byte unsigned integer 16-битное целое беззнаковое

//adUnsignedTinyInt 17 1-byte unsigned integer 8-битное целое беззнаковое

//adUserDefined 132 User-defined variable Пользовательский тип

//adVarBinary 204 Binary value Двоичные значения

//adVarChar 200 String

//adVariant 12 Automation variant

//adVarNumeric 139 Variable width exact numeric with signed scale

//adVarWChar 202 Null-terminated Unicode character string Какие-то строки в юникоде

//adWChar 130 Null-terminated Unicode character string Какие-то строки в юникоде

Возврат Новый Структура("ИмяПараметра, Тип, Размер, Значение",

ИмяПараметра, Тип, Размер, Значение);

КонецФункции

// Возвращает COMОбъект("ADODB.Connection") для ПолучитьРезультатЗапроса параметр Connection.

// Параметры:

// ДанныеДляПодключения - структура - данные для подключения,см. СоздатьДанныеДляПодключения

// Отказ - булево - при ошибке в выполнении функции- истина

// ОписаниеОшибки - строка - описание ошибки выполнения функции

//

// Возвращаемое значение:

// Структура - COMОбъект("ADODB.Connection")

//

Функция НовоеСоединениеССУБД(ДанныеДляПодключения, Отказ = Ложь, ОписаниеОшибки = Неопределено) Экспорт

Connection = Новый COMОбъект("ADODB.Connection");

Connection.ConnectionTimeOut = 30;

Connection.CommandTimeout = 220;

Если ДанныеДляПодключения.ТипСУБД = "MSSQL" Тогда

Connection.Provider = "SQLOLEDB";

Connection.ConnectionString = "Data Source="

+ ДанныеДляПодключения.АдресСервера + ";Initial Catalog="

+ ДанныеДляПодключения.ИмяБД

+ ";Persist Security Info=True;User ID="

+ ДанныеДляПодключения.Логин

+ ";Password=" + ДанныеДляПодключения.Пароль;

ИначеЕсли ДанныеДляПодключения.ТипСУБД = "PGSQL" Тогда

Connection.Provider = "MSDASQL.1";

Connection.ConnectionString = "DRIVER={PostgreSQL Unicode};Data Source="

+ ДанныеДляПодключения.DataSource + ";SERVER="

+ ДанныеДляПодключения.АдресСервера + ";PORT="

+ ДанныеДляПодключения.Порт+";DATABASE="

+ ДанныеДляПодключения.ИмяБД

+ ";UID=" + ДанныеДляПодключения.Логин

+ ";PWD=" + ДанныеДляПодключения.Пароль+";STMT=" + "utf8";

КонецЕсли;

Попытка

Connection.Open();

Исключение

ОписаниеОшибки = ОписаниеОшибки();

Connection = Неопределено;

Отказ = Истина;

КонецПопытки;

Возврат Connection;

КонецФункции

// Возвращает Строку Base64

// Параметры:

// safeArray - COMSafeArray

//

// Возвращаемое значение:

// Строка - строка, закодированную по алгоритму base64

//

// Используется для adLongVarBinary 205 Long binary value

// На паример, для получения данных типа image sql

//

Функция ВСтрокуBase64(safeArray) Экспорт

obj = Новый COMОбъект("MSXML2.DomDocument.3.0");

helper=obj.createElement("a");

helper.DataType = "bin.base64";

helper.nodeTypedValue = safeArray;

Возврат helper.text;

КонецФункции

// Закрывает соединение с СУБД.

// Параметры:

// Соединение - COMОбъект - COMОбъект("ADODB.Connection")

// Отказ - булево - если передается истина, процедура не выполняется, при ошибке в выполнении процедуры- истина

// ОписаниеОшибки - строка - описание ошибки выполнения процедуры

//

Процедура ЗакрытьСоединениеССУБД(Соединение, Отказ = Ложь, ОписаниеОшибки = Неопределено) Экспорт

Если Не Отказ Тогда

Попытка

Если Соединение <> Неопределено Тогда

Соединение.Close();

КонецЕсли;

Исключение

Отказ = Истина;

ОписаниеОшибки = ОписаниеОшибки();

КонецПопытки;

КонецЕсли;

Соединение = Неопределено;

КонецПроцедуры

#КонецОбласти

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

#Область Подключение\_к\_SQL

// Устанавливает параметры запроса.

// Параметры:

// Command - COMОбъект -("ADODB.Command") коннект к бд.

// Параметры - структура, массив структур - структура содержит значение параметра:

// ИмяПараметра - строка - имя парамета

// Тип - число - тип параметра

// Размер - число - размер значения

// Значение - любое - значение параметра.

//

Процедура УстановитьПараметрыЗапроса(Command, Параметры)

Если ТипЗнч(Параметры) = Тип("Структура") Тогда

ДбавитьПараметр(Command, Параметры);

Иначе

Для Каждого Параметр Из Параметры Цикл

ДбавитьПараметр(Command, Параметр);

КонецЦикла;

КонецЕсли;

КонецПроцедуры

Процедура ДбавитьПараметр(Command, Параметр)

Param1 = Command.CreateParameter(Параметр.ИмяПараметра, Параметр.Тип, Параметр.Размер);

Command.Parameters.Append(Param1);

Param1.Value = Параметр.Значение;

КонецПроцедуры

Функция SQLТипВ1СТип(Num, size)

ПараметрыСтроки = Новый КвалификаторыСтроки(size);

Если Num = 20 Тогда

Возврат Новый ОписаниеТипов("Число");

ИначеЕсли Num = 128 Тогда

Возврат Новый ОписаниеТипов("Булево");

ИначеЕсли Num = 8 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки);

ИначеЕсли Num = 136 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки);

ИначеЕсли Num = 129 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки);

ИначеЕсли Num = 7 Тогда

Возврат Новый ОписаниеТипов("Дата");

ИначеЕсли Num = 133 Тогда

Возврат Новый ОписаниеТипов("Дата");

ИначеЕсли Num = 134 Тогда

Возврат Новый ОписаниеТипов("Дата");

ИначеЕсли Num = 135 Тогда

Возврат Новый ОписаниеТипов("Дата");

ИначеЕсли Num = 14 Тогда

Возврат Новый ОписаниеТипов("Число");

ИначеЕсли Num = 0 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adEmpty 0 no value

ИначеЕсли Num = 3 Тогда

Возврат Новый ОписаниеТипов("Число"); // adInteger 3 4 - byte signed integer

ИначеЕсли Num = 205 Тогда

Возврат Неопределено; // adLongVarBinary 205 Long binary value

ИначеЕсли Num = 201 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adLongVarChar 201 Long string value

ИначеЕсли Num = 203 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adLongVarWChar 203 Long Null - terminates string value

ИначеЕсли Num = 131 Тогда

Возврат Новый ОписаниеТипов("Число"); // adNumeric 131 Number with fixed precision and scale

ИначеЕсли Num = 5 Тогда

Возврат Новый ОписаниеТипов("Число");

ИначеЕсли Num = 4 Тогда

Возврат Новый ОписаниеТипов("Число"); // adSingle 4 Single - precision floating - point value

ИначеЕсли Num = 2 Тогда

Возврат Новый ОписаниеТипов("Число"); // adSmallInt 2 2 - byte signed integer

ИначеЕсли Num = 16 Тогда

Возврат Новый ОписаниеТипов("Число"); // adTinyInt 16 1 - byte signed integer

ИначеЕсли Num = 21 Тогда

Возврат Новый ОписаниеТипов("Число"); // adUnsignedBigInt 21 8 - byte unsigned integer

ИначеЕсли Num = 19 Тогда

Возврат Новый ОписаниеТипов("Число"); // adUnsignedInt 19 4 - byte unsigned integer

ИначеЕсли Num = 18 Тогда

Возврат Новый ОписаниеТипов("Число"); // adUnsignedSmallInt 18 2 - byte unsigned integer

ИначеЕсли Num = 17 Тогда

Возврат Новый ОписаниеТипов("Число"); // adUnsignedTinyInt 17 1 - byte unsigned integer

ИначеЕсли Num = 132 Тогда

Возврат Новый ОписаниеТипов("Число"); // adUserDefined 132 User - defined variable

ИначеЕсли Num = 204 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adVarBinary 204 Binary value

ИначеЕсли Num = 12 Тогда

Возврат Новый ОписаниеТипов("Число"); // adVariant 12 Automation variant

ИначеЕсли Num = 139 Тогда

Возврат Новый ОписаниеТипов("Число"); // adVarNumeric 139 Variable width exact numeric with signed scale

ИначеЕсли Num = 202 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adVarWChar 202 Null - terminated Unicode character string

ИначеЕсли Num = 130 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки); // adWChar 130

ИначеЕсли Num = 200 Тогда

Возврат Новый ОписаниеТипов("Строка", , , , ПараметрыСтроки);

Иначе

Возврат Новый ОписаниеТипов("Строка");

КонецЕсли;

КонецФункции

#КонецОбласти

#Область Конструктор\_запросов\_в\_SQL

Функция СобратьДобавляемыеСтроки(Строки)

МассивСтрок = Новый Массив;

Для Каждого ДанныеСтроки Из Строки Цикл

Строка = Новый Массив;

Строка.Добавить("(");

Строка.Добавить(ДанныеСтроки);

Строка.Добавить(")");

МассивСтрок.Добавить(СтрСоединить(Строка));

КонецЦикла;

Возврат СтрСоединить(МассивСтрок, ", ");

КонецФункции

#КонецОбласти

#КонецОбласти

**Примеры использования**

**Получений данных из MS SQL**

SQLЗапрос = "

| SELECT TOP (1) [Photo]

| FROM [Personnel].[dbo].[tblPersons]

| Where [PFR\_Nom] = ? and not [Photo] is null

| order by [Person\_Id] desc";

ПараметрыЗапроса = Новый Массив();

ПараметрыЗапроса.Добавить(КОРВ\_РаботаСБазамиSQL.СоздатьПараметрЗапроса("PFR\_Nom",,,"123456789"));

Фото = КОРВ\_РаботаСБазамиSQL.ВыполнитьЗапросВ\_Personnel(SQLЗапрос, ПараметрыЗапроса);

Если Фото <> Неопределено и Фото.Количество() > 0 Тогда

КартинкаBase64 = КОРВ\_РаботаСБазамиSQL.ВСтрокуBase64(Фото[0].Photo); //Получаю фото в Base64 для дальнейшей обработки

КонецЕсли;

**Запись данных в PostgreSQL**

Строки = Новый Массив;

Строки.Добавить("'DDD', 30, 30, 2, '20210201'");

Строки.Добавить("'YYY', 30, 30, 2, '20210201'");

ТекстЗапроса = КОРВ\_РаботаСБазамиSQL.ЗапросИнсерт("public.weather", "city, temp\_lo, temp\_hi, prcp, date", Строки);

ДанныеДляПодключения = КОРВ\_РаботаСБазамиSQL.СоздатьДанныеДляПодключения("172.20.3.252", "testdb", "test", "12345678", "PGSQL",, "PostgreSQL35W");

Connection = КОРВ\_РаботаСБазамиSQL.НовоеСоединениеССУБД(ДанныеДляПодключения);

КОРВ\_РаботаСБазамиSQL.ВыполнитьЗапрос(ТекстЗапроса,, Connection);

КОРВ\_РаботаСБазамиSQL.ЗакрытьСоединениеССУБД(Connection);

**Добавить, или обновить записи в PostgreSQL при записи регистра сведений**

Процедура ПриЗаписи(Отказ, Замещение)

ДанныеДвиженийДляСинхронизации = Новый Массив;

Для Каждого ЗаписьРС Из ЭтотОбъект Цикл

ДобавитьДанныеДвиженияДляЗаписиВSchedules(ЗаписьРС, ДанныеДвиженийДляСинхронизации);

КонецЦикла;

ТекстЗапроса = SchedulesДобавитьИлиОбновить(ДанныеДвиженийДляСинхронизации);

ВыполнитьЗапросВПроходную(ТекстЗапроса);

КонецПроцедуры

// Возвращает строку для запроса с добавляемымы данными.

//

// Параметры:

// ЗаписьРС - запись регистра - запись для рс КОРВ\_ДокументыПроходной

// МассивСтрок - массив - массив куда вставляются строки для SQL

//

Процедура ДобавитьДанныеДвиженияДляЗаписиВSchedules(ЗаписьРС, МассивСтрок) Экспорт

СобраннаяСтрока = Новый Массив;

СобраннаяСтрока.Добавить(ФорматЧислаPG(ЗаписьРС.ГрафикРаботы.id));

СобраннаяСтрока.Добавить(ФорматДатыPG(ЗаписьРС.Дата));

СобраннаяСтрока.Добавить(ФорматВремяPG(ЗаписьРС.ВремяПрихода));

СобраннаяСтрока.Добавить(ФорматВремяPG(ЗаписьРС.ВремяУхода));

СобраннаяСтрока.Добавить(ФорматВремяPG(ЗаписьРС.ОбедВремяПрихода));

СобраннаяСтрока.Добавить(ФорматВремяPG(ЗаписьРС.ОбедВремяУхода));

МассивСтрок.Добавить(СтрСоединить(СобраннаяСтрока, ", "));

КонецПроцедуры

// Возвращает строку запроса для добавления набора значений, или обновлении при наличие в таблицу DocPass.

//

// Параметры:

// Строки - массив строк - в каждом элементе массива находится строка со значениями полей по порядку, указанному в параметре "Поля".

// Пример: Строки = Новый Массив;

// Строки.Добавить("'DDD', 30, 30, 2, '20210201'");

// Строки.Добавить("'YYY', 30, 30, 2, '20210201'");

// ОбновляемыеПоля - строка - поля для обновление в случае, если найдена запись по ключу

// По умолчанию: "date = VALUES(date), time\_begin = VALUES(time\_begin),

// time\_end = VALUES(time\_end), free\_sign = VALUES(free\_sign),

// info = VALUES(info)"

//

Функция SchedulesДобавитьИлиОбновить(Строки) Экспорт

Поля = "id, date, time\_begin, time\_end, dinner\_begin, dinner\_end";

КлючевоеПоле = "id, date";

Возврат КОРВ\_РаботаСБазамиSQL.ЗапросДобавитьИлиОбновить\_Pg\_SQL("public.""Schedules""", Поля, Строки,ПолучитьОбновляемыеПоляТаблицы(Поля, КлючевоеПоле), КлючевоеПоле);

КонецФункции

// Возвращает результат выполнения запроса.

//

// Параметры:

// SQLText - текст - текст запроса.

// ПолучитьРезультат - булево - при значении истина возвращает результат запрос

// Параметры - массив структур - см. КОРВ\_РаботаСБазамиSQL.СоздатьПараметрЗапроса().

//

// Возвращаемое значение:

// COMОбъект - COMОбъект c результатом запроса

//

Функция ВыполнитьЗапросВПроходную(SQLText, ПолучитьРезультат = Ложь, Параметры = Неопределено) Экспорт

Отказ = Ложь;

//Если это копия базы, то соеденяется с тестовой базой проходной

Если РегламентныеЗаданияСервер.РаботаСВнешнимиРесурсамиЗаблокирована() Тогда

Connection = СоединениеСБДПроходнойТЕСТ();

Иначе

Connection = СоединениеСБДПроходной();

КонецЕсли;

Результат = Неопределено;

ОписаниеОшибки = "";

Если ПолучитьРезультат Тогда

Результат = КОРВ\_РаботаСБазамиSQL.ПолучитьРезультатЗапроса(SQLText,, Connection, Параметры,, Отказ, ОписаниеОшибки);

Иначе

КОРВ\_РаботаСБазамиSQL.ВыполнитьЗапрос(SQLText,, Connection, Параметры,, ОписаниеОшибки, Отказ);

КонецЕсли;

КОРВ\_РаботаСБазамиSQL.ЗакрытьСоединениеССУБД(Connection);

Если Отказ Тогда

ЗаписьЖурналаРегистрации("Ошибка выполнения запроса в проходную", УровеньЖурналаРегистрации.Ошибка,,, ОписаниеОшибки);

КонецЕсли;

Возврат Результат;

КонецФункции

// Возвращает строку с полями таблицы для обновления в запросе КОРВ\_РаботаСБазамиSQL.ЗапросДобавитьИлиОбновить\_Pg\_SQL().

//

// Параметры:

// ПоляТаблицы - строка - поля таблицы через запятую

// Ключи - строка - ключевые поля таблицы через запятую

//

// Возвращаемое значение:

// строка - срока обновления полей

//

Функция ПолучитьОбновляемыеПоляТаблицы(ПоляТаблицы, Ключи)

МасивПолей = СтрРазделить(СтрЗаменить(ПоляТаблицы, " ", ""), ",", Ложь);

МасивКлючей = СтрРазделить(СтрЗаменить(Ключи, " ", ""), ",", Ложь);

МассивПолейЗамены = Новый Массив;

Для Каждого Поле Из МасивПолей Цикл

Если МасивКлючей.Найти(Поле) = Неопределено Тогда

МассивПолейЗамены.Добавить("" + Поле + " = excluded." + Поле);

КонецЕсли;

КонецЦикла;

Возврат СтрСоединить(МассивПолейЗамены, ", ");

КонецФункции

**Удалить данный из SQL**

Процедура КОРВ\_ПриЗаписи(Отказ, Замещение)

ТекстЗапроса = ScheduleLinksУдалить(ЭтотОбъект.Отбор.Регистратор.Значение);

ВыполнитьЗапросВПроходную(ТекстЗапроса);

КонецПроцедуры

Функция ScheduleLinksУдалить(Документ) Экспорт

Возврат КОРВ\_РаботаСБазамиSQL.ЗапросДелет("public.""ScheduleLinks""", "id = " + Документ.id + " and id\_doc\_type = " + ПолучитьТипДокументаВПроходной(Документ));

КонецФункции

Функция ПолучитьТипДокументаВПроходной(Документ)

Возврат 1;

КонецФункции

// Возвращает результат выполнения запроса.

//

// Параметры:

// SQLText - текст - текст запроса.

// ПолучитьРезультат - булево - при значении истина возвращает результат запрос

// Параметры - массив структур - см. КОРВ\_РаботаСБазамиSQL.СоздатьПараметрЗапроса().

//

// Возвращаемое значение:

// COMОбъект - COMОбъект c результатом запроса

//

Функция ВыполнитьЗапросВПроходную(SQLText, ПолучитьРезультат = Ложь, Параметры = Неопределено) Экспорт

Отказ = Ложь;

//Если это копия базы, то соеденяется с тестовой базой проходной

Если РегламентныеЗаданияСервер.РаботаСВнешнимиРесурсамиЗаблокирована() Тогда

Connection = СоединениеСБДПроходнойТЕСТ();

Иначе

Connection = СоединениеСБДПроходной();

КонецЕсли;

Результат = Неопределено;

ОписаниеОшибки = "";

Если ПолучитьРезультат Тогда

Результат = КОРВ\_РаботаСБазамиSQL.ПолучитьРезультатЗапроса(SQLText,, Connection, Параметры,, Отказ, ОписаниеОшибки);

Иначе

КОРВ\_РаботаСБазамиSQL.ВыполнитьЗапрос(SQLText,, Connection, Параметры,, ОписаниеОшибки, Отказ);

КонецЕсли;

КОРВ\_РаботаСБазамиSQL.ЗакрытьСоединениеССУБД(Connection);

Если Отказ Тогда

ЗаписьЖурналаРегистрации("Ошибка выполнения запроса в проходную", УровеньЖурналаРегистрации.Ошибка,,, ОписаниеОшибки);

КонецЕсли;

Возврат Результат;

КонецФункции