Документация

SQL-скрипт

Компонент предназначен для выполнения пользовательских SQL-запросов в различных СУБД и может быть полноценно вставлен в поток обработки данных. Поддерживается выполнение скриптов, которые не возвращают курсоры, и работа с транзакциями. Для работы компонента необходимо Подключение к соответствующей базе данных, по аналогии с компонентами Импорт из базы данных и Экспорт в базу данных. Компонент может принимать на вход табличные данные и переменные, которые можно использовать в скрипте как параметры и макроподстановки.

Порты

Вход

- **Подключение** — принимает параметры подключения к базе данных.
- **Входной набор данных** (необязательный) — таблица с данными для параметризации скрипта. При подключении скрипт выполняется для каждой строки набора.
- **Управляющие переменные** (необязательный) — переменные для настройки компонента или использования в скрипте.

> **Важно**: При конфликте имен полей набора данных и переменных приоритет отдается данным из набора.

Выход

- **Выходной набор данных** — таблица с ошибками, которая содержит три колонки:
  - номер строки данных, на которой произошла ошибка;
  - код завершения;
  - сообщение об ошибке.

Мастер настройки

Мастер настройки содержит следующие параметры:

Подключение — позволяет активировать соединение с СУБД для выполнения скриптов.
Выполнять скрипт для каждой строки (включен по умолчанию) — если входной набор данных подключен, скрипт выполняется для каждой строки. В противном случае — однократно.

Разбивать на команды — выполняет скрипт как последовательность отдельных команд, если СУБД не поддерживает SQL-запросы с множественными командами. При получении подтверждения от драйвера базы данных о таком ограничении, параметр автоматически активируется и блокируется для изменений.
Тайм-аут работы скрипта (секунды) — определяет максимальное время выполнения:

- в режиме построчного выполнения — тайм-аут применяется к обработке каждой строки данных;
- при разбивке скрипта на команды — ограничение действует на весь набор команд.

Значение тайм-аута может задаваться через переменные.
Игнорировать ошибки — продолжать выполнение после ошибки (только для режима «одна строка = одна транзакция»).
Периодичность фиксации транзакции (строк) — задаёт интервал, с которым скрипт разбивает выполнение на отдельные транзакции. Есть возможность настроить периодичность транзакций — использовать шаблоны и задать период вручную или через переменные.

Режимы	Управление транзакциями
Управляется скриптом (0)	Транзакции объявляются и закрываются в скрипте
Весь набор данных (-1)	Все строки в одной транзакции. Ошибка отменяет весь набор.
Каждую строку (1)	Каждая строка в отдельной транзакции. Ошибка вызывает откат только текущей строки.
На блок (N>0)	Блок из N строк в одной транзакции. Ошибка отменяет весь блок.

Окно мастера настройки разделено на 3 области:

- В левой верхней части находится вкладка *Таблица/представления*. После установки соединения с СУБД отображаются объекты базы данных: схемы, таблицы и представления. Поля таблиц можно добавлять в скрипт методом перетаскивания *Drag-and-drop* (по умолчанию перетаскивается непосредственное имя объекта, а при зажатой клавише CTRL — полное имя) или с помощью команд контекстного меню: *Вставить имя в редактор SQL кода* или *Вставить полное имя в редактор SQL кода*.

- Под областью *Таблица/представления* отображаются вкладки *Поля входного набора и Переменные*, в которых выводятся списки полей и переменных, передаваемых на вход узла. Поля и переменные можно перетащить методом *Drag-and-drop* (по умолчанию перетаскивается параметр, а при зажатой клавише CTRL — макроподстановка) или вставить в окно редактирования скрипта с помощью команд контекстного меню: *Параметр* и *Подстановка*. Если на входных портах поля и переменные не заданы, то эта область мастера настройки будет находится в свернутом состоянии.

  > **Примечание**: Если имя переменной совпадает с именем поля таблицы, система выводит предупреждение о конфликте имен.

- В правой части располагается окно *Редактор кода*, которое является основным редактором SQL-запросов. Возможности редактора: автодополнение кода, выделение ключевых слов, подсветка различных SQL-конструкций и перетаскивание объектов из базы данных, входных данных и переменных с помощью *Drag-and-drop*.

Особенности работы с транзакциями и SQL-скриптами

MS SQL Server — если в скрипте есть SELECT-запрос, который возвращает много данных, то последующие команды могут не сработать. Обработка таких наборов данных не производится. Это ограничение касается только подключений к MS SQL через ODBC драйвер на Linux.

В **SQLite** транзакция существует в контексте соединения с базой данных. Если транзакция начата (например, через `BEGIN`), но не завершена командами `COMMIT` или `ROLLBACK` до закрытия соединения, то соединение сохраняет состояние с активной (незавершённой) транзакцией. При попытке начать новую транзакцию на том же соединении возникнет ошибка. Для того чтобы отменить все операции транзакции и вернуть базу данных в состояние до начала транзакции, требуется выполнить команду `ROLLBACK`. Для того чтобы сохранить все выполненные в транзакции операции, требуется выполнить команду `COMMIT`.

Firebird:

- Транзакции из скрипта не поддерживаются. При попытке выполнить команду `SET TRANSACTION` возникнет ошибка.
- Несколько команд выполняются через `EXECUTE BLOCK`.
- DDL-команды возможны только через `EXECUTE IMMEDIATE [динамический SQL-скрипт]`. Для фиксации изменений используется `with autonomous transaction`.

MySQL:

- Значения параметров передаются не бинарно, а подставляются в текст запроса.
- Остановка скрипта после `SELECT` может не сработать.

Oracle:

- Выполнение несколько команд осуществляется через блок `BEGIN..END`.
- Выполнение DDL-команды возможно только через `EXECUTE IMMEDIATE [динамический SQL-скрипт]`.

PostgreSQL:

- Если в скрипте открывается транзакция, но до её закрытия возникает ошибка, то сессия PostgreSQL становится неработоспособной (активация узлов с этим *Подключением* завершается ошибкой). Чтобы вывести подключение из неработоспособного состояния, нужно выполнить `ROLLBACK` (отменяет все операции транзакции и возвращает базу данных в состояние до начала транзакции) или `COMMIT` (сохраняет все выполненные в транзакции операции).
- Используется текстовый протокол Simple Query.