DataOps.onETL

Unit#2

Коннекторы к БД: какие они бывают, какие методы предоставляют

Саттар Гюльмамедов

РО команды ETL

МТС Тета x DataOps Platform

onETL урок # 2

→ Типы коннекторов

→ Как их импортировать

→ Где выполняются методы коннектора

→ Зачем нужны коннекторы

→ Чем коннекторы отличаются
от объектов чтения/записи

→ Логирование

Виды коннекторов

→ DBConnection

→ FileConnection

→ FileDataFrameConnection

import

```python
from onetl.connection import  ...
```

Коннекторы...

→ Обычные

→ Необычные

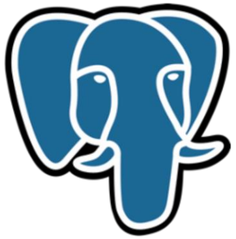
«Обычные» DBConnection

→ MSSQL

→ Postgres

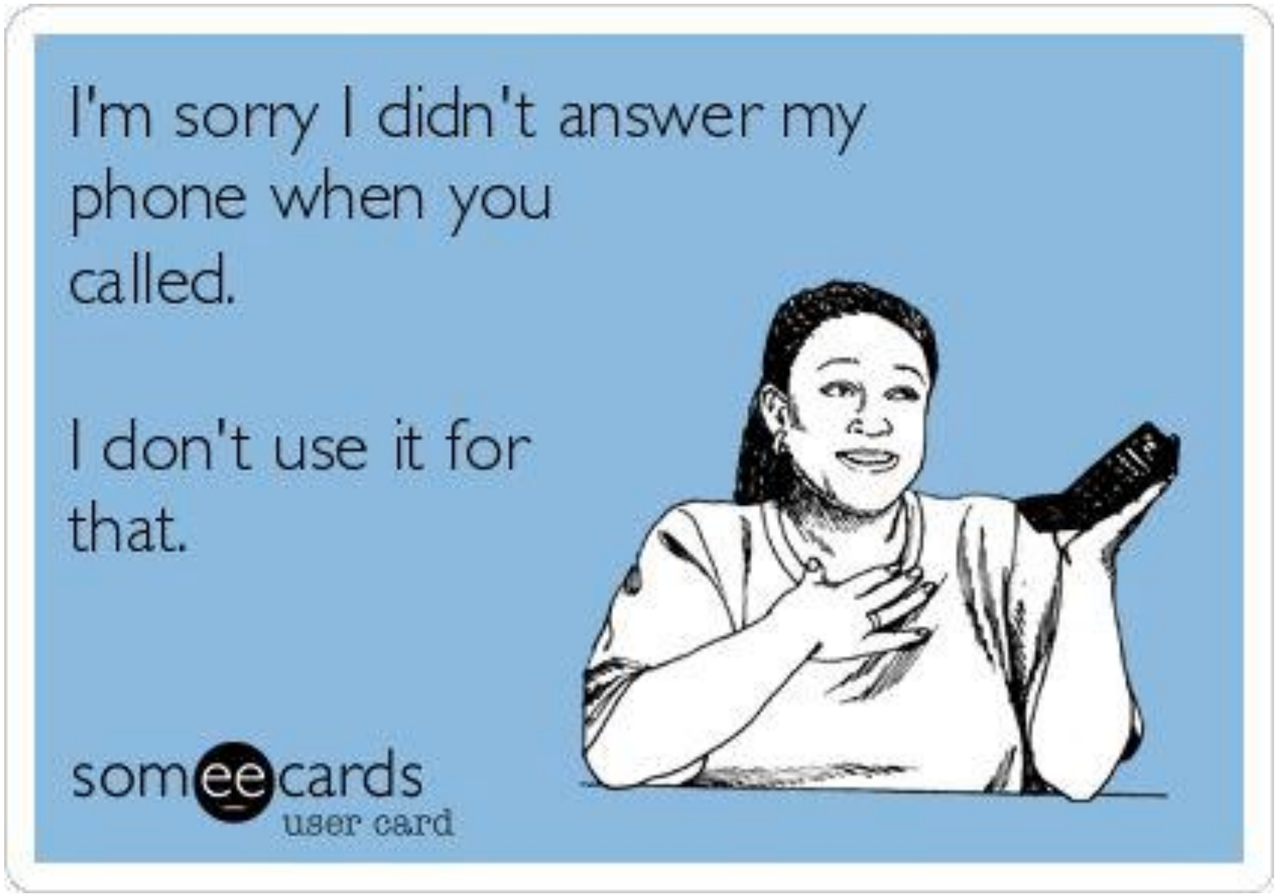
→ MySQL

→ Oracle



Логирование

https://onetl.readthedocs.io/en/stable/logging.html


I'm sorry I didn't answer my phone when you called.
I don't use it for that.
someecards
user card

Функции логирования

→ setup_logging()

→ setup_clients_logging()

→ set_default_logging_format()

setup_logging()

Параметры:

→ уровень логирования

→ активация логирования на клиентах

Устанавливает:

→ обработчик stderr

→ формат «2023-05-31 11:22:33.456 [INFO] MainThread: message»

→ корневой уровень логирования

→ уровень логирования onETL

→ уровень логирования клиентов

setup_clients_logging()

Параметры:

→ уровень логирования

Влияет на:

→ ftputil

→ hdfs

→ minio

→ paramiko

→ py4j

→ pyspark

→ webdav3

set_default_logging_format()

→ Устанавливает формат логирования вида
«2023-05-31 11:22:33.456 [INFO] MainThread: message»

Предназначено для использования в IDE или процессах ETL

Настройка логирования

Python

```python
from onetl.log import setup_logging


setup_logging()
```

Документация коннекторов: MSSQL

https://onetl.readthedocs.io/en/stable/connection/db_connection/mssql/connection.html

Документация коннекторов: PostgreSQL

https://onetl.readthedocs.io/en/stable/connection/db_connection/postgres/connection.html

Документация коннекторов: MySQL

https://onetl.readthedocs.io/en/stable/connection/db_connection/mysql/connection.html

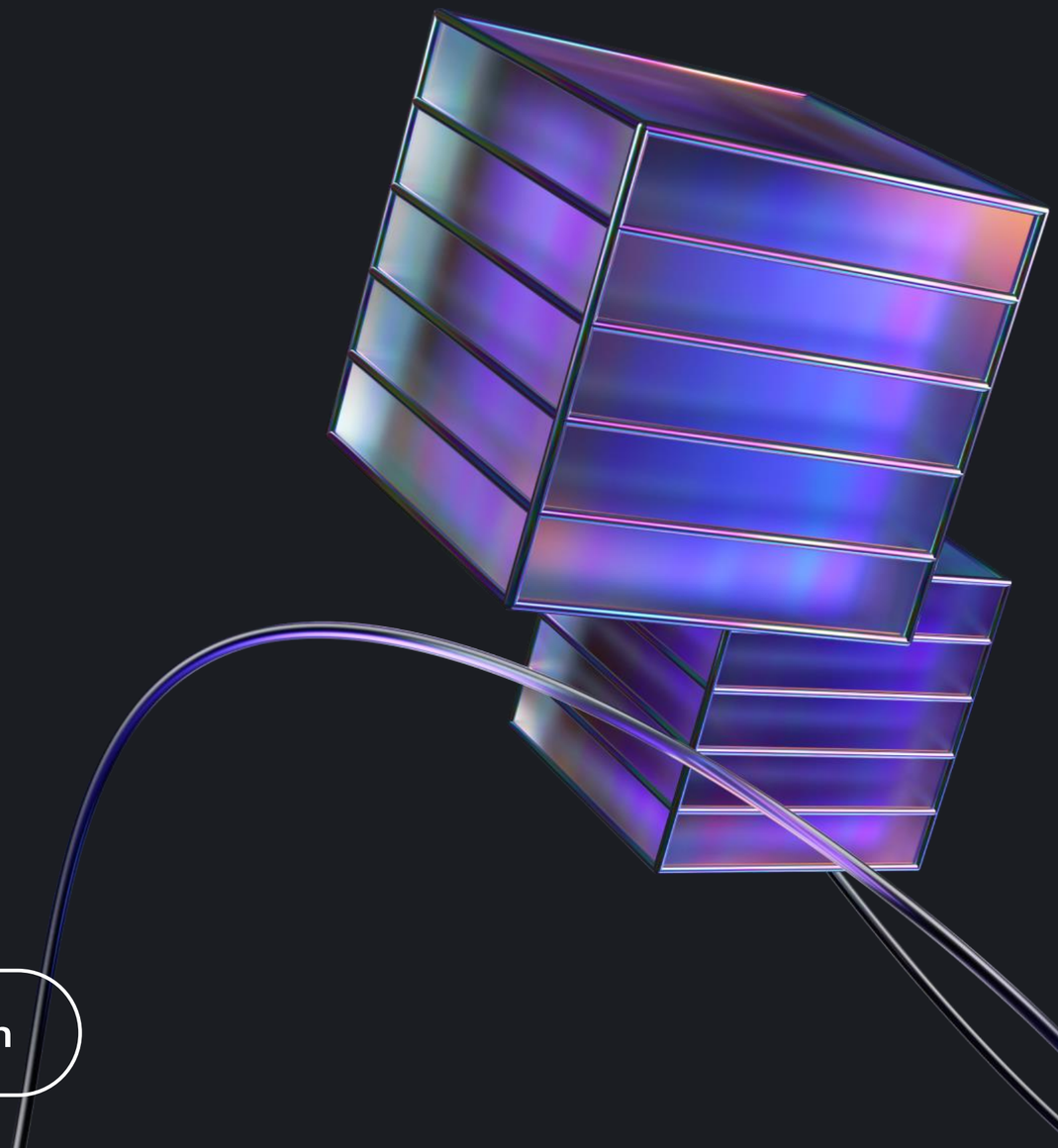
Документация коннекторов: Oracle

https://onetl.readthedocs.io/en/stable/connection/db_connection/oracle/connection.html

Конструктор

	mssql	postgres	mysql	oracle
spark	SparkSession	SparkSession	SparkSession	SparkSession
user	str	str	str	str
password	SecretStr	SecretStr	SecretStr	SecretStr
host	Host	Host	Host	Host
port	1433	5432	3306	1521
database	str	str	Optional[str] = None	
				service_name: Optional[str] = None
				sid: Optional[str] = None
extra	MSSQLExtra = MSSQLExtra()	PostgresExtra = PostgresExtra()	MySQLExtra = MySQLExtra(useUnicode ='yes', characterEncoding='UTF-8')	OracleExtra = OracleExtra()

extra:



МТС Тета x DataOps Platform

extra:
MSSQL

https://learn.microsoft.com/en-us/sql/connect/jdbc/setting-the-connection-properties#properties

extra:
PostgreSQL

https://github.com/pgjdbc/pgjdbc#connection-properties

extra:
MySQL

https://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html

extra:
Oracle

https://docs.oracle.com/en/database/oracle/oracle-database/23/jajdb/oracle/jdbc/OracleConnection.html

Методы коннектора

→ check()

→ get_packages()

→ sql()

→ fetch()

→ execute()

→ close()

check()

MSSQL:

```SQL
SELECT 1 AS field
```

Oracle:

```SQL
SELECT 1 FROM dual
```

MySQL и PostgreSQL:

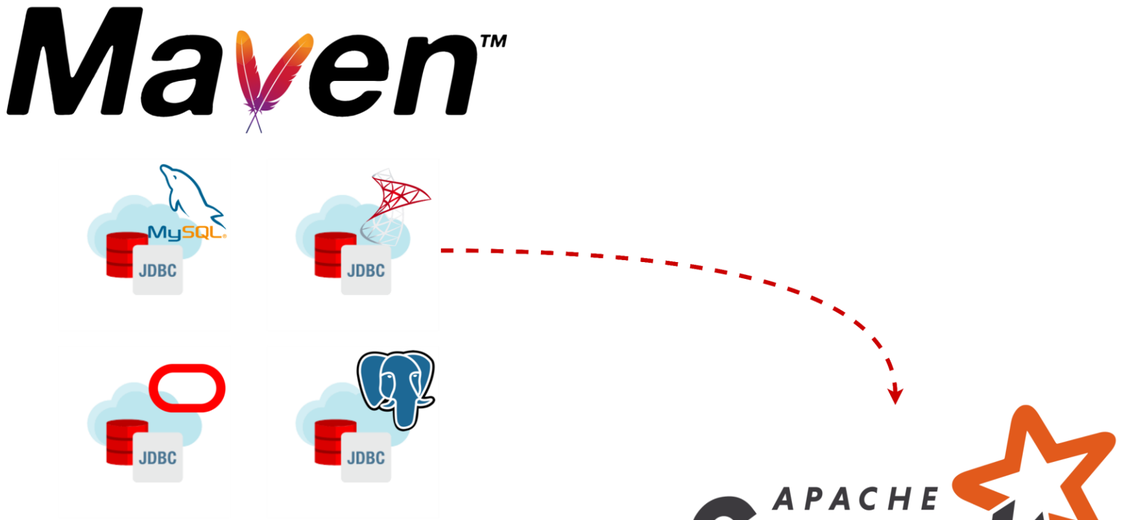
```SQL
SELECT 1
```

Результат check()

Python

```
<DBMS_type>(user='<user_name>',
            password=SecretStr('**********'),
            host='host_ip_address',
            database='<database_name>',
            port=<port_number>,
            extra=Extra(<param_name> = '<param_value>'))
```

get_packages()



ivysettings.xml

XML

```xml
<ivysettings>
    <settings defaultResolver="main"/>
    <resolvers>
      <chain name="main" returnFirst="true">
          <!-- Use Maven cache -->
          <ibiblio name="local-maven-cache" m2compatible="true" root="file://${user.home}/.m2/repository"/>
          <!-- Use ~/.ivy2/jars/*.jar files -->
          <ibiblio name="local-ivy2-cache" m2compatible="false" root="file://${user.home}/.ivy2/jars"/>
          <!-- BigData packages -->
          <ibiblio name="bigdata" m2compatible="true" root="https://artifactory.mts.ru/artifactory/maven-bigdata/" />
          <!-- Nexus package proxy -->
          <ibiblio name="nexus-proxy-maven" m2compatible="true" root="https://nexus.services.mts.ru/repository/maven-central/" />
          <ibiblio name="nexus-proxy-apache" m2compatible="true" root="https://nexus.services.mts.ru/repository/maven-apache/" />
          <ibiblio name="nexus-proxy-jcenter" m2compatible="true" root="https://nexus.services.mts.ru/repository/maven-jcenter/" />
      </chain>
    </resolvers>
</ivysettings>
```

Spark-сессия с ivysettings

Python

```python
from pyspark.sql import SparkSession  # импортируем установленный Spark
from onetl.connection import Postgres
from onetl.log import setup_logging

# настраиваем формат логов и уровень логирования
setup_logging()

# создаем сессию
spark = (
  SparkSession.builder
  .config("spark.master", "local[*]")  # локальный запуск, используются все доступные ядра (между [] можно указать их количество)
  .config("spark.appName", "mysessionname")  # название сессии
  .config("spark.jars.packages", ",".join(Postgres.get_packages()))  # перечисляем пакеты для подключения к конкретному типу БД
  .config("spark.jars.ivySettings", "…/ivysettings.xml")  # путь к ivysettings.xml
  .getOrCreate()
)
```

https://confluence.mts.ru/x/5CUgKw#id-ДокументацияonETL-Влюбомдругомокружении(накорпоративномноутбуке,вVDI,навиртуалкезаBalabit,вdockerобразе)

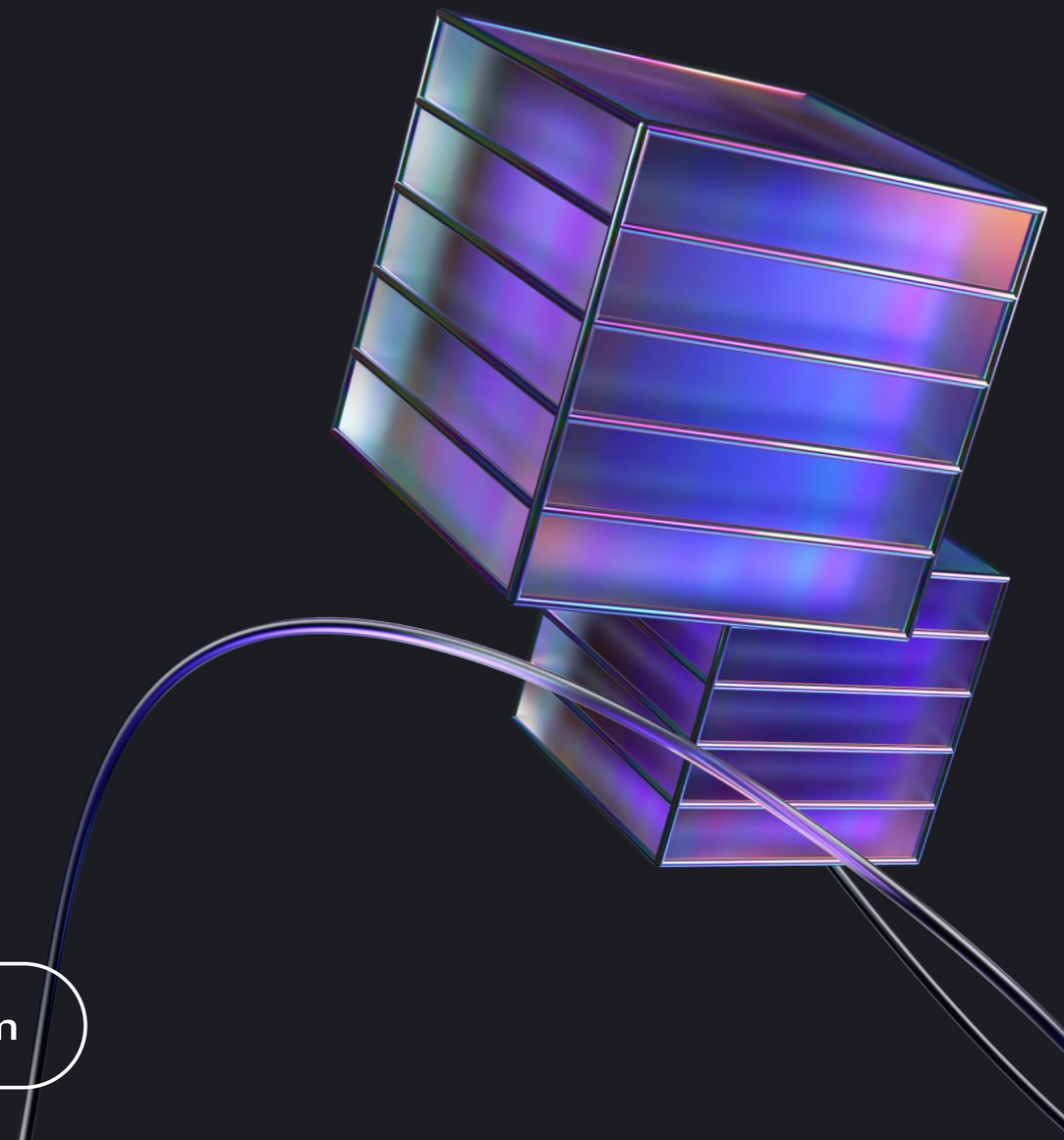
*только для внутренних сотрудников компании МТС

close()

Read & Write Options

https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

Демо Spark JDBC To Other Databases



МТС Тета x DataOps Platform

Опции чтения

option	Умолчание	Возможные значения
partitioning_mode	range	range, hash, mod
partition_column	умолчания нет	
num_partitions	1	
lower_bound	умолчания нет	
upper_bound	умолчания нет	
session_init_statement	умолчания нет	
query_timeout	может быть установлено jdbc-драйвером	
fetchsize	10 000	

Опции записи

option	Умолчание	Возможные значения
query_timeout	может быть установлено jdbc-драйвером	
fetchsize	может быть установлено jdbc-драйвером	
if_exists	append	append, replace_entire_table, ignore, error
batchsize	20 000	
isolation_level	READ_UNCOMMITTED	NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE

sql()

→ Выполняется на executor

→ Параметры

1. query
   - только SELECT
   - применяется синтаксис конкретной СУБД
   - SHOW не поддерживается
2. options
   - ReadOptions

→ Возвращает DataFrame

fetch()

→ Выполняется на driver

→ Параметры

1. query
   - SELECT + SHOW
   - применяется синтаксис конкретной СУБД
2. options
   - ReadOptions

→ Возвращает DataFrame

execute()

→ Выполняется на driver

→ Параметры

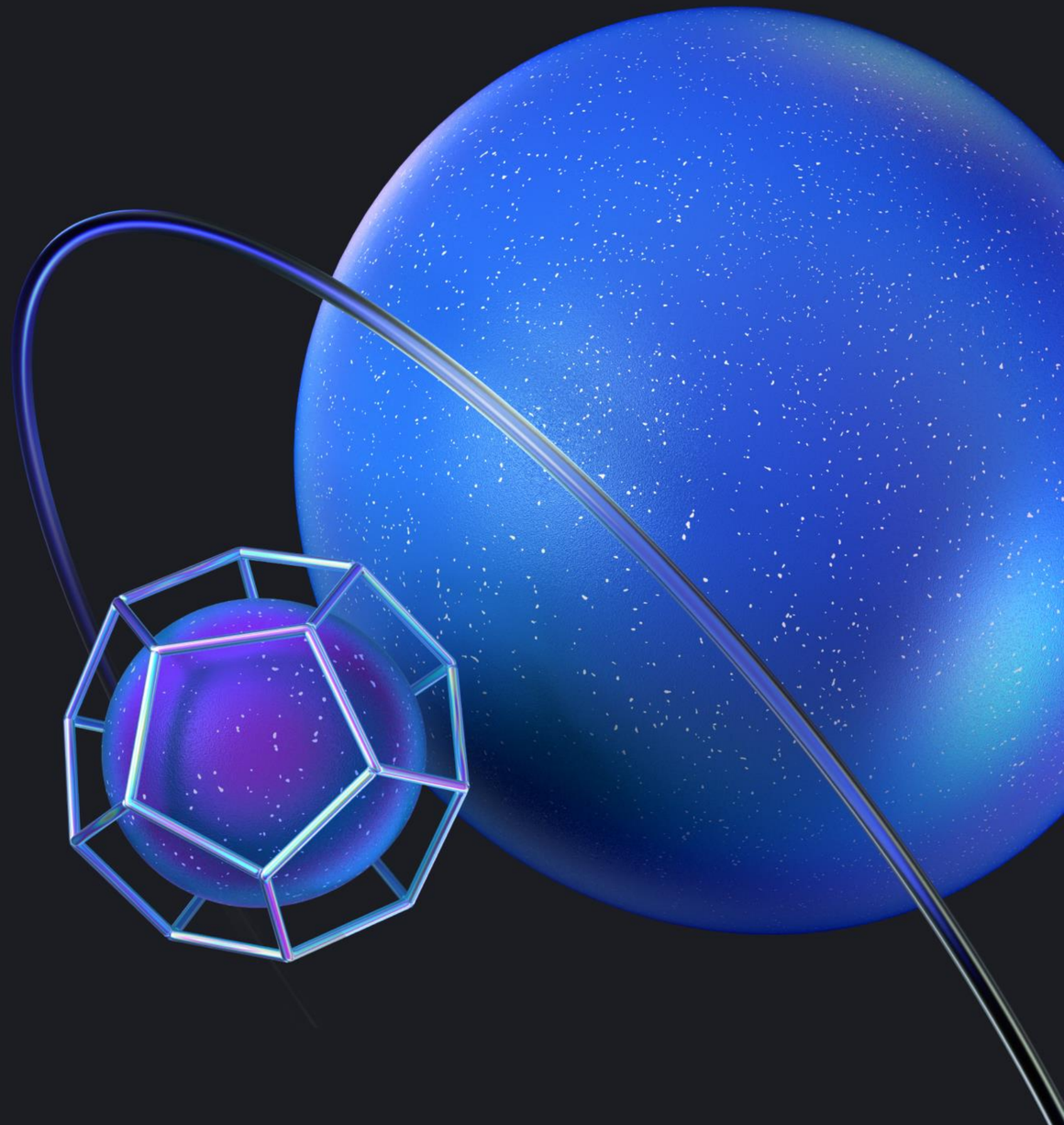
1. query
   - поддерживается все, кроме SELECT
   - применяется синтаксис конкретной СУБД
2. options
   - ReadOptions

→ Возвращает DataFrame
(только в случае инструкции RETURNING или вызова процедуры)

Отличия от DBReader/DBWriter

Функциональность	Коннекторы	Объекты чтения/записи
Установка соединения	+	-
Методы для driver	+	-
Методы для executor	+	+
Использование стратегий	-	+
Модуль	onetl.connection	onetl.db

Спасибо!



onetools@mts.ru
https://t.me/c/1511728757/5

МТС Тета x DataOps Platform