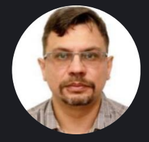
DataOps.onETL

Unit#5

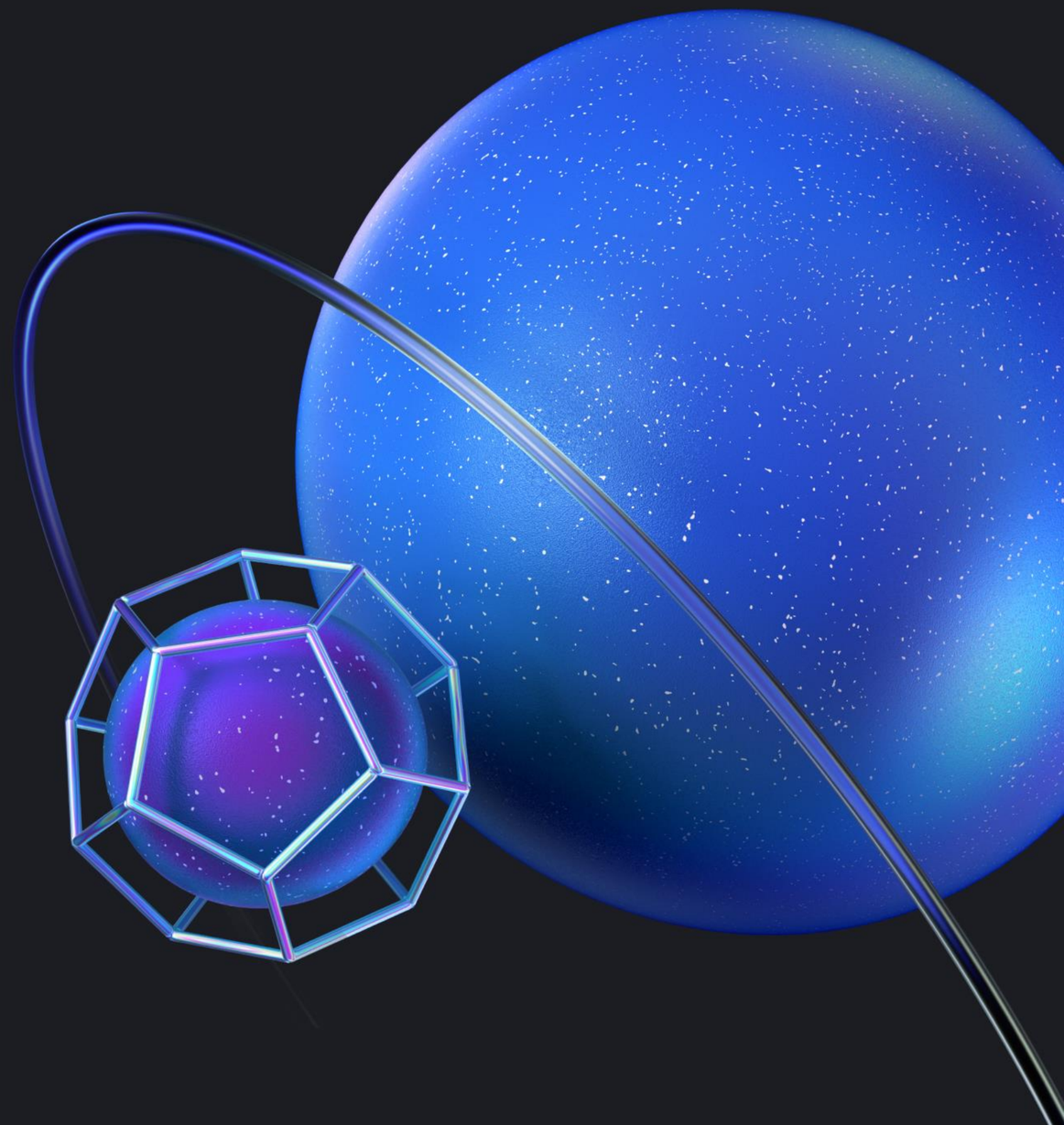
Нестандартные коннекторы: Greenplum, MongoDB, Teradata



Саттар Гюльмамедов

PO команды ETL

МТС Тета x DataOps Platform



onETL урок # 5

- → Нестандартные коннекторы
- → Hive
- → Greenplum
- → Clickhouse
- → Kafka
- → MongoDB
- → Teradata

МТС Тета x DataOps Platform

Нестандартные коннекторы

- → Hive
- → Greenplum
- → Clickhouse
- → Kafka
- → MongoDB
- → Teradata

Hive

→ Hive()

→ check()

→ get_current()

→ sql()

→ execute()

МТС Тета x DataOps Platform

Конструктор Hive

```python
from onetl.connection import Hive
from pyspark.sql import SparkSession

spark = SparkSession.builder \
                    .appName("spark-app-name") \
                    .enableHiveSupport() \
                    .getOrCreate()

hive = Hive(cluster="rnd-dwh", spark=spark).check()
```

Параметр	Тип значения	Объяснение
spark	SparkSession	сессия Spark
cluster	str	имя кластера

Hive+DBReader

Python

```
...

connection = Hive(cluster="rnd-dwh", spark=spark)

reader = DBReader(
    connection=connection,
    source="default.test",
    where="d_id > 100",
    columns=["d_id", "d_name", "d_age"]
)

df = reader.run()
```

Hive+DBWriter

Параметр	Тип и допустимые значения	Объяснение
if_exists	append, replace_overlapping_partitions, replace_entire_table, ignore, error	Поведение при записи в существующую таблицу
format	str, **default**: `orc`	Формат записываемых файлов
partition_by	[str]	Список столбцов, используемых для партиционирования. **None** отключает партиционирование.
bucket_by	(int, [str])	Количество бакетов и имена столбцов. **None** означает, что бакеты не используются.
sort_by	[str]	Каждый файл в бакете будет отсортирован по этим столбцам. **None** означает, что сортировка не используется.
compression	str	Алгоритм сжатия используемый при создании файлов на HDFS.

Hive методы

```python
...

connection = Hive(cluster="rnd-dwh", spark=spark)
df = connection.sql("SELECT * FROM mytable")
connection.execute("ALTER TABLE mytable DROP PARTITION(date='2023-02-01')")

...
```

→ execute()

→ sql()

Greenplum

→ Greenplum()

→ get_packages()

→ check()

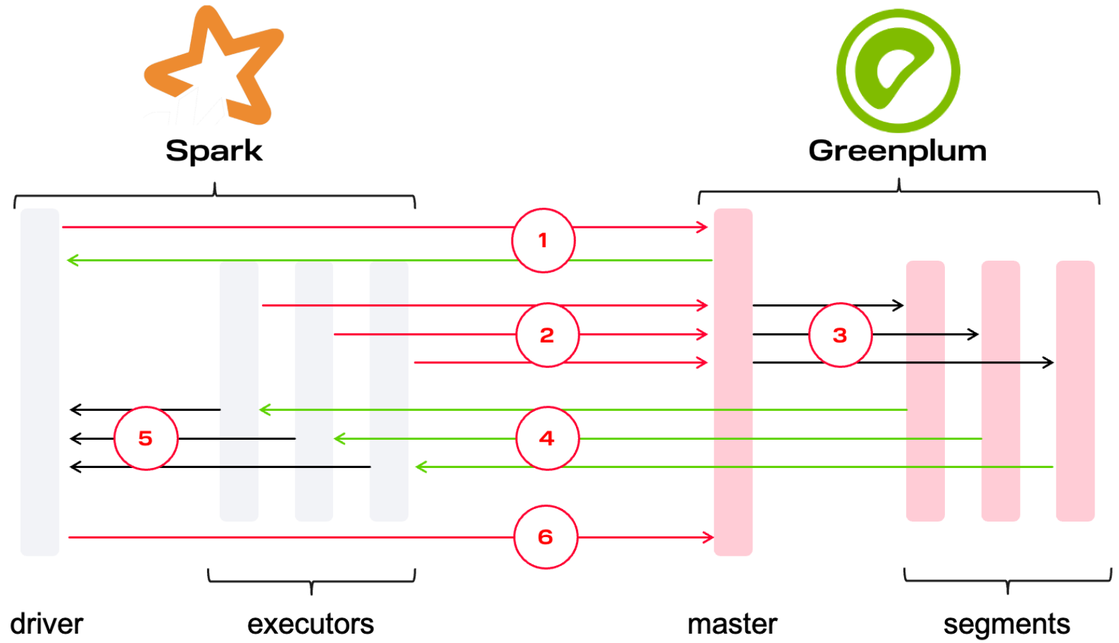
→ execute()

→ fetch()

→ close()

МТС Тета x DataOps Platform

Взаимодействие Spark и GP



1. Установка соединения
2. Запрос данных
3. Распределение задач
4. Возврат данных
5. Данные получены
6. Завершение соединения

МТС Тета x DataOps Platform

Конструктор Greenplum

```python
from onetl.connection import Greenplum
from pyspark.sql import SparkSession

maven_packages = Greenplum.get_packages(spark_version="3.2")
spark = (
    SparkSession.builder.appName("spark-app-name")
    .config("spark.jars.packages", ",".join(maven_packages))
    .config("spark.executor.allowSparkContext", "true")
    .config("spark.dynamicAllocation.maxExecutors", 10)
    .config("spark.executor.cores", 1)
    .getOrCreate()
)

extra = {
    "server.port": "41000-42000",
}

greenplum = Greenplum(
    host="master.host.or.ip",
    user="user",
    password="*****",
    database="target_database",
    extra=extra,
    spark=spark,
)
```

Параметр	Тип значения	Комментарий
host	str	Адрес мастер-ноды кластера Greenplum
port	int, default: 5432	Порт для подключения
user	str	Имя пользователя
password	str	Пароль пользователя
database	str	Имя БД для подключения
spark	SparkSession	Сессия Spark
extra	dict	Дополнительные параметры

PG JDBC Options

GP Connector Options

МТС Тета x DataOps Platform

Greenplum+DBReader

```python
from onetl.connection import Greenplum
from onetl.db import DBReader

greenplum = Greenplum( ... )

read_options = Greenplum.ReadOptions(
    partition_column="reg_id",
    num_partitions=10,
)

reader = DBReader(
    connection=greenplum,
    source="schema.table",
    columns=["id", "key", "CAST(value AS string) value", "updated_dt"],
    where="key = 'something'",
    options=read_options
)

df = reader.run()
```

МТС Тета x DataOps Platform

Greenplum+DBWriter

```python
from onetl.connection import Greenplum
from onetl.db import DBWriter

greenplum = Greenplum( ... )

df = ...

write_options = Greenplum.WriteOptions(
    if_exists="append",
    truncate="false",
    distributedBy="mycolumn",
)

writer = DBWriter(
    connection=greenplum,
    target="schema.table",
    options=write_options
)

writer.run(df)
```

Параметр	Тип значения	Объяснение
if_exists	append, replace_overlapping_partitions, replace_entire_table, ignore, error	Поведение при записи в существующую таблицу
truncate	str	Флаг перезаписи данных для управления подходом к очистке таблицы
distributedBy	[str]	Список столбцов, используемых для управления распределением данных по нодам-сегментам

МТС Тета x DataOps Platform

Greenplum методы

```python
...
connection.execute("CREATE TABLE target_table(id NUMBER, data VARCHAR)")
...
df = connection.fetch("SELECT * FROM mytable", {"fetchsize": 10000})
...
connection.close()
...
```

→ execute()

→ fetch()

→ sql()

Clickhouse

→ Clickhouse()

→ get_packages()

→ check()

→ sql()

→ execute()

→ fetch()

→ close()

МТС Тета x DataOps Platform

Конструктор Clickhouse

```python
from onetl.connection import Clickhouse
from pyspark.sql import SparkSession

maven_packages = Clickhouse.get_packages()
spark = (
    SparkSession.builder.appName("spark-app-name")
    .config("spark.jars.packages", ",".join(maven_packages))
    .getOrCreate()
)

clickhouse = Clickhouse(
    host="database.host.or.ip",
    user="user",
    password="*****",
    extra={"continueBatchOnError": "false"},
    spark=spark,
)
```

Параметр	Тип значения	Комментарий
host	str	Адрес узла Clickhouse
port	int, default: 8123	Порт для подключения
user	str	Имя пользователя
password	str	Пароль пользователя
database	str	Имя БД для подключения
spark	SparkSession	Сессия Spark
extra	dict	Дополнительные параметры

Clickhouse+DBReader

```python
from onetl.connection import Clickhouse
from onetl.db import DBReader

clickhouse = Clickhouse( ... )

reader = DBReader(
    connection=clickhouse,
    source="schema.table",
    columns=["id", "key", "CAST(value AS String) value", "updated_dt"],
    where="key = 'something'",
    options=Clickhouse.ReadOptions(partition_column="id", num_partitions=10),
)
df = reader.run()
```

Spark JDBC Options

Clickhouse datatypes

МТС Тета x DataOps Platform

Clickhouse+DBWriter

```python
from onetl.connection import Clickhouse
from onetl.db import DBWriter

clickhouse = Clickhouse( ... )

df = ...

writer = DBWriter(
    connection=clickhouse,
    target="schema.table",
    options=Clickhouse.WriteOptions(
        if_exists="append",
        createTableOptions="ENGINE = MergeTree() ORDER BY id",
    ),
)

writer.run(df)
```

Параметр	Тип значения	Объяснение
if_exists	append, replace_overlapping_partitions, replace_entire_table, ignore, error	Поведение при записи в существующую таблицу
batchsize	int, default: 20 000	Максимальное количество строк в пакете записи данных
isolation_level	str, default: 'READ_UNCOMMITTED' NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE	Уровень изоляции транзакции

МТС Тета x DataOps Platform

Clickhouse методы

→ execute()

→ fetch()

→ sql()

→ close()

Python

```python
clickhouse = Clickhouse( ... )
df = clickhouse.sql(
    """
    SELECT
        id,
        key,
        CAST(value AS String) value,
        updated_at
    FROM
        some.mytable
    WHERE
        key = 'something'
    """,
    options=Clickhouse.ReadOptions(
        partition_column="id",
        num_partitions=10,
        lower_bound=0,
        upper_bound=1000,
    ),
)

clickhouse.close()
```

Python

```python
...


with connection:
    connection.execute("CREATE TABLE mytable(id NUMBER, data VARCHAR)")
    connection.execute("INSERT INTO mytable VALUES (1, 'test record')",
                       {"isolationLevel": "READ_COMMITTED"})
    ...
    df = connection.fetch("SELECT * FROM mytable")


...
```

Kafka

```python
...

kafka = Kafka(
    addresses=["mybroker:9092", "anotherbroker:9092"],
    cluster="my-cluster",
    protocol=Kafka.SSLProtocol(
        keystore_type="PEM",
        keystore_certificate_chain=Path("path/to/user.crt").read_text(),
        keystore_key=Path("path/to/user.key").read_text(),
        truststore_type="PEM",
        truststore_certificates=Path("/path/to/server.crt").read_text(),
    ),
    auth=Kafka.ScramAuth(
        user="me",
        password="abc",
        digest="SHA-512",
    ),
    spark=spark,
)
```

- → Kafka()
- → get_packages()
- → check()
- → close()

МТС Тета x DataOps Platform

Kafka Protocol

Реализации:

→ PlaintextProtocol

→ SSLProtocol

Методы:

→ get_options()

→ cleanup()

МТС Тета x DataOps Platform

Kafka SSLProtocol

- → keystore_type
- → keystore_location
- → keystore_password
- → keystore_certificate_chain
- → keystore_key
- → key_password
- → truststore_type
- → truststore_location
- → truststore_password
- → truststore_certificates

МТС Тета x DataOps Platform

Kafka Auth

Реализации:

→ BasicAuth

→ KerberosAuth

→ ScramAuth

Методы:

→ get_jaas_conf()

→ get_options()

→ cleanup()

Basic

Scram

Kerberos

Kafka+DBReader

```python
schema = StructType(
    [
        StructField("value", BinaryType(), nullable=True),
        StructField("key", BinaryType(), nullable=True),
        StructField("topic", StringType(), nullable=False),
        StructField("partition", IntegerType(), nullable=False),
        StructField("offset", LongType(), nullable=False),
        StructField("timestamp", TimestampType(), nullable=False),
        StructField("timestampType", IntegerType(), nullable=False),
        # возвращается только если include_headers=True
        StructField(
            "headers",
            ArrayType(
                StructType(
                    [
                        StructField("key", StringType(), nullable=False),
                        StructField("value", BinaryType(), nullable=True),
                    ],
                ),
            ),
            nullable=True,
        ),
    ],
)
```

Kafka+DB Reader

Spark Kafka Integration

МТС Тета x DataOps Platform

Kafka+DBWriter

```python
schema = StructType(
    [
        # Обязательное поле
        StructField("value", BinaryType(), nullable=True),
        # Эти поля могут быть опущены
        StructField("key", BinaryType(), nullable=True),
        StructField("partition", IntegerType(), nullable=True),
        # передается только в случае include_headers=True
        StructField(
            "headers",
            ArrayType(
                StructType(
                    [
                        StructField("key", StringType(), nullable=False),
                        StructField("value", BinaryType(), nullable=True),
                    ],
                ),
            ),
            nullable=True,
        ),
    ],
)
```

Kafka+DB Writer

Spark Kafka Integration

МТС Тета x DataOps Platform

MongoDB

→ MongoDB()

→ get_packages()

→ check()

→ pipeline()

MongoDB()

```python
from onetl.connection import MongoDB
from pyspark.sql import SparkSession

maven_packages = MongoDB.get_packages(spark_version="3.4")
spark = (
    SparkSession.builder.appName("spark-app-name")
    .config("spark.jars.packages", ",".join(maven_packages))
    .getOrCreate()
)

mongo = MongoDB(
    host="master.host.or.ip",
    user="user",
    password="*****",
    database="target_database",
    spark=spark,
)
```

Параметр	Тип значения	Комментарий
host	str	Адрес хоста MongoDB
port	int, default: 27017	Порт для подключения
user	str	Имя пользователя
password	str	Пароль пользователя
database	str	Имя БД для подключения
spark	SparkSession	Сессия Spark
extra	dict	Дополнительные параметры

MongoDB+DBReader

```python
...
from onetl.connection import MongoDB
from onetl.db import DBReader

from pyspark.sql.types import (
    StructType,
    StructField,
    IntegerType,
    StringType,
    TimestampType,
)

mongodb = MongoDB( ... )

# mandatory
df_schema = StructType(
    [
        StructField("_id", StringType()),
        StructField("some", StringType()),
        StructField(
            "field",
            StructType(
                [
                    StructField("nested", IntegerType()),
                ],
            ),
        ),
        StructField("updated_dt", TimestampType()),
    ]
)

reader = DBReader(
    connection=mongodb,
    source="some_collection",
    df_schema=df_schema,
    where={"field": {"$eq": 123}},
    hint={"field": 1},
    options=MongoDBReadOptions(batchSize=10000),
)
df = reader.run()
```

МТС Тета x DataOps Platform

MongoDB+DBWriter

```python
...
from onetl.connection import MongoDB
from onetl.db import DBWriter

mongodb = MongoDB( ... )

df = ...

writer = DBWriter(
    connection=mongodb,
    target="schema.table",
    options=MongoDB.WriteOptions(
        if_exists="append",
    ),
)

writer.run(df)
```

Параметр	Тип значения	Объяснение
if_exists	append, replace_entire_collection, ignore, error	Поведение при записи в существующую таблицу

MongoDB: pipeline()

```python
...
from pyspark.sql.types import (
    DoubleType,
    IntegerType,
    StringType,
    StructField,
    StructType,
    TimestampType,
)

df_schema = StructType(
    [
        StructField("_id", IntegerType()),
        StructField("some_string", StringType()),
        StructField("some_int", IntegerType()),
        StructField("some_datetime", TimestampType()),
        StructField("some_float", DoubleType()),
    ],
)

df = connection.pipeline(
    collection="collection_name",
    df_schema=df_schema,
    pipeline={"$match": {"some_int": {"$gt": 999}}},
    options=MongoDB.PipelineOptions(hint={"_id": 1})
)
...
```

Parameter	Value	Comment
collection	str	Имя коллекции
pipeline	list[dict]	Применяемый pipeline
df_schema	StructType, default: None	Схема получаемых данных
options	PipelineOptions	Дополнительные опции pipeline

MongoDB Pipeline Syntax

Pipeline, Read Options

Teradata

- → Teradata()
- → get_packages()
- → check()
- → sql()
- → fetch()
- → execute()
- → close()

МТС Тета x DataOps Platform

Teradata()

```python
from onetl.connection import Teradata
from pyspark.sql import SparkSession

maven_packages = Teradata.get_packages()
spark = (
    SparkSession.builder.appName("spark-app-name")
    .config("spark.jars.packages", ",".join(maven_packages))
    .getOrCreate()
)

teradata = Teradata(
    host="database.host.or.ip",
    user="user",
    password="*****",
    extra={
        "TMODE": "TERA",
        "LOGMECH": "LDAP",
        "LOG": "TIMING",
    },
    spark=spark,
)
```

Параметр	Тип значения	Комментарий
host	str	Адрес хоста Teradata
port	int, default: 1025	Порт для подключения
user	str	Имя пользователя
password	str	Пароль пользователя
database	str	Имя БД для подключения
spark	SparkSession	Сессия Spark
extra	dict	Дополнительные параметры

Teradata+DBReader

Python

```python
from onetl.connection import Teradata
from onetl.db import DBReader

teradata = Teradata( ... )

reader = DBReader(
    connection=teradata,
    source="database.table",
    columns=["id", "key", "CAST(value AS VARCHAR) value", "updated_dt"],
    where="key = 'something'",
    options=Teradata.ReadOptions(
        partition_column="id",
        num_partitions=10,
        partitioning_mode="hash",
    ),
)
df = reader.run()
```

Teradata+DBWriter

```python
from onetl.connection import Teradata
from onetl.db import DBWriter

teradata = Teradata( ... )

df = ...

writer = DBWriter(
    connection=teradata,
    target="database.table",
    options=Teradata.WriteOptions(
        if_exists="append",
        createTableOptions="NO PRIMARY INDEX",
    ),
)

writer.run(df.repartition(1))
```

Параметр	Тип значения	Объяснение
if_exists	append, replace_entire_table, ignore, error	Поведение при записи в существующую таблицу
batchsize	int, default: 20 000	Максимальное количество строк в одном пакете данных
isolation_level	str, default: *READ_UNCOMMITTED*	Уровень изоляции транзакции
query_timeout		Количество секунд в течение которого ожидается выполнение запроса

МТС Тета x DataOps Platform

Teradata методы

```python
df = connection.sql("SELECT * FROM mytable")
df = teradata.fetch(
    "SELECT value FROM some.reference_table WHERE key = 'some_constant'",
    options=Teradata.FetchOptions(query_timeout=10),
)
teradata.execute("DROP TABLE database.table")
connection.close()
```

- → fetch()
- → execute()
- → close()
- → sql()

execute

fetch

sql

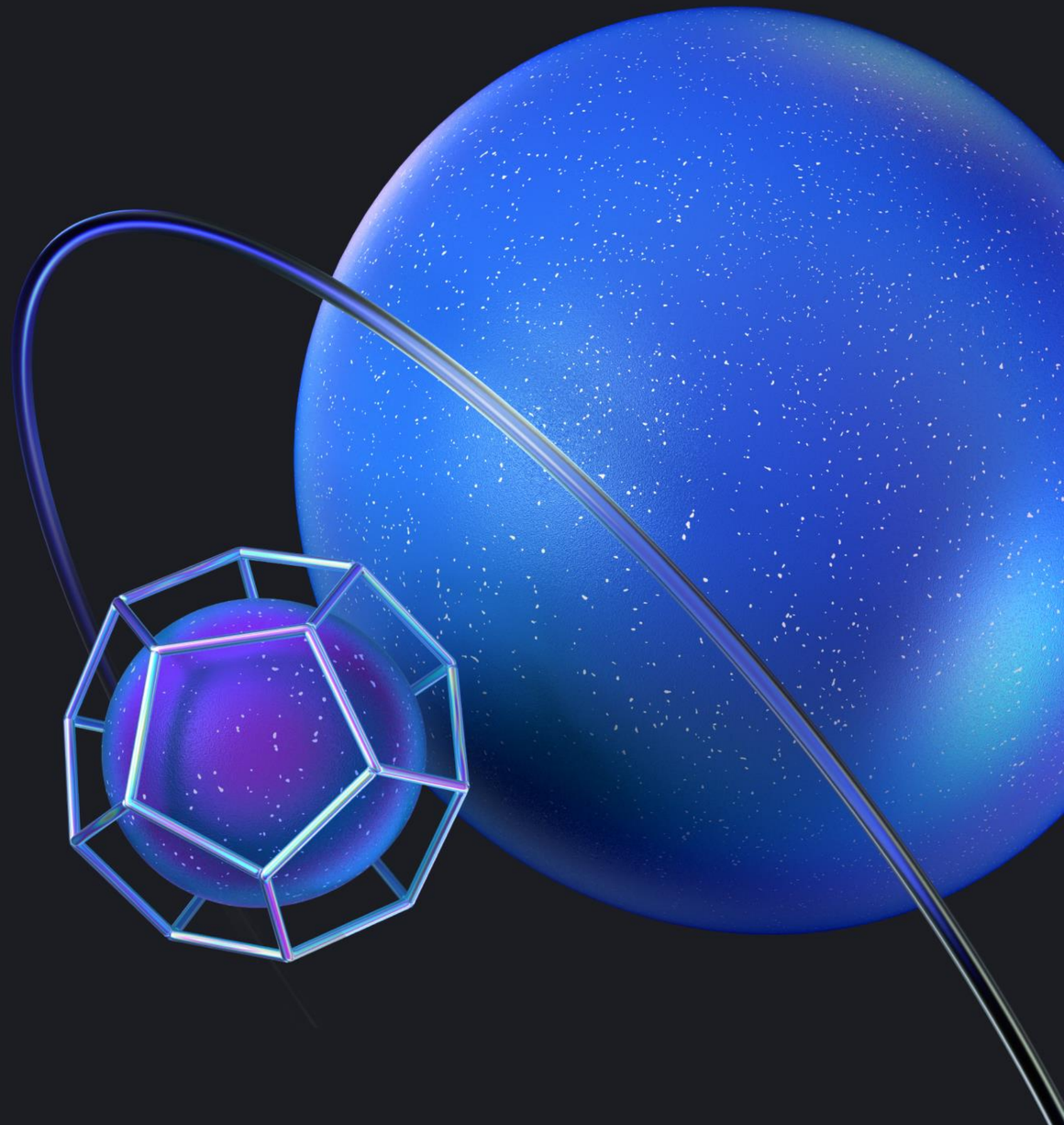
МТС Тета x DataOps Platform

onETL урок # 5

- → Нестандартные коннекторы
- → Hive
- → Greenplum
- → Clickhouse
- → Kafka
- → MongoDB
- → Teradata

МТС Тета x DataOps Platform

Спасибо!



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

МТС Тета x DataOps Platform