Отчет по DolphinDB

* 1. История развития СУБД  
     Страна происхождения: Китай
     1. В 2015 году команда разработчиков начала работу над DolphinDB.
     2. В 2017 году была выпущена первая версия базы данных.
     3. В 2018 году DolphinDB стала доступна для коммерческого использования.

DolphinDB используется для аналитики и обработки потоков данных в реальном времени.

* 1. Инструменты для взаимодействия с СУБД

API и другие методы для взаимодействия:

* + 1. JDBC
    2. JSON over HTTP
    3. Kafka
    4. MQTT (Message Queue Telemetry Transport)
    5. ODBC
    6. OPC DA
    7. OPC UA
    8. RabbitMQ
    9. WebSocket

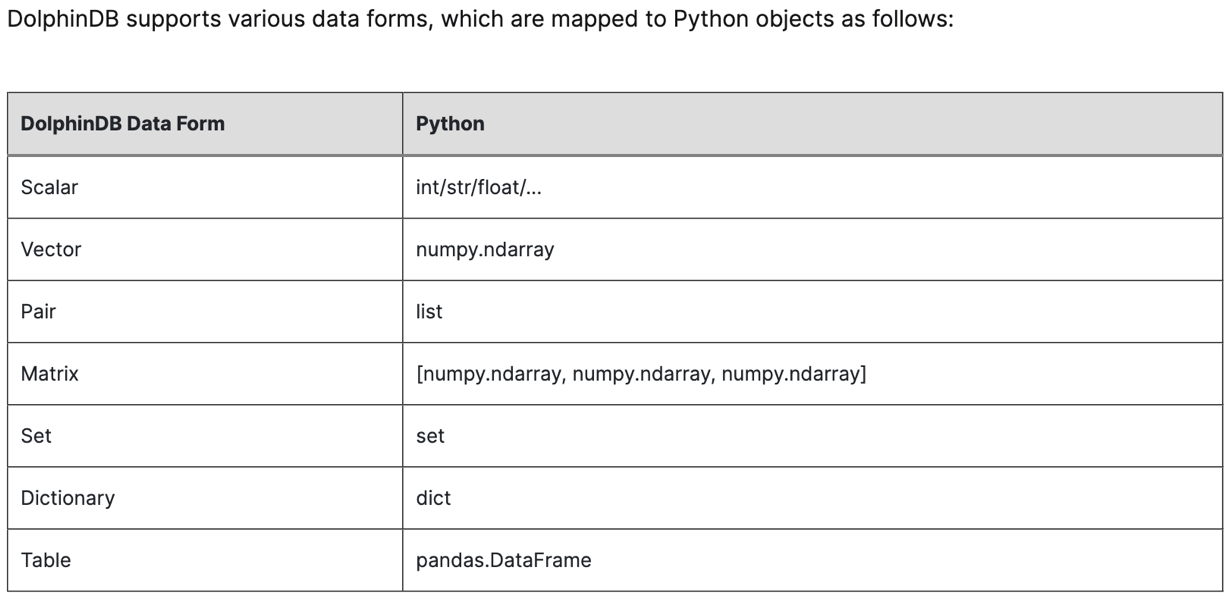
Поддерживаемые языки программирования:

* + 1. C#
    2. C++
    3. Go
    4. Java
    5. JavaScript
    6. MatLab
    7. Python
    8. R
    9. Rust
  1. Какой database engine используется в вашей СУБД?  
     До версии 2.0 Dolphin DB единственным средством хранения данных был OLAP-движок. Каждый столбец в разделе таблицы сохраняется в виде файла. Данные хранятся в том порядке, в котором они записываются, что делает запись очень эффективной.

В версии 2.0 был добавлен TSDB. Он разработан на основе логически структурированного дерева слияния (LSM Tree). Данные в каждом разделе хранятся в файлах уровней. Данные в каждом файле уровней сортируются и индексируются блоками

Также есть поддержка In-Memory OLTP

* 1. Как устроен язык запросов в вашей СУБД? Разверните БД с данными и выполните ряд запросов.

SQL-like query language  


Я установил сервер DolphinDB на свой компьютер с Windows. Взаимодействие с БД буду проводить с помощью Python API

PS C:\DolphinDB\_Win64\_V3.00.0.1\server> ./dolphindb

DolphinDB Systems 3.00.0.1 64 bit Copyright (c) 2011~2024 DolphinDB, Inc. Licensed to Trial Users. Expires on 2042.01.01 (Build:2024.04.24)ild:2024.04.24)

PS C:\DolphinDB\_Win64\_V3.00.0.1\server> python

Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import dolphindb as ddb

>>> a = ddb.session()

>>> s.connect('localhost', 8848)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 's' is not defined

>>> a.connect('localhost', 8848)

True

**Scalar**

>>> s.upload({'scalar\_sample': 1})

62776640

>>> s.run("typestr(scalar\_sample);")

'LONG'

>>> s.run("scalar\_sample;")

1

**Vector**

>>> s.upload({'vector\_sample': np.array([1, 3])})

65583680

>>> s.run("typestr(vector\_sample);")

'FAST LONG VECTOR'

>>> s.run("vector\_sample;")

array([1, 3])

**Matrix**

>>> s.upload({'matrix\_sample': np.array([[1, 2, 3], [4, 5, 6]])})

65484832

>>> s.run("typestr(matrix\_sample);")

'FAST LONG MATRIX'

>>> s.run("matrix\_sample;")

[array([[1, 2, 3],

[4, 5, 6]]), None, None]

**Set**

>>> s.upload({'set\_sample': {1, 4, 7}})

65578432

>>> s.run("typestr(set\_sample);")

'LONG SET'

>>> s.run("set\_sample;")

{1, 4, 7}

**Dictionary**

>>> s.upload({'dict\_sample': {'a': 1}})

58318576

>>> s.run("typestr(dict\_sample);")

'STRING->LONG DICTIONARY'

>>> s.run("dict\_sample;")

{'a': 1}

**Table**

>>> df = pd.DataFrame({'a': [1, 2, 3], 'b': ['a', 'b', 'c']})

>>> s.upload({'table\_sample': df})

63409760

>>> s.run("typestr(table\_sample);")

'IN-MEMORY TABLE'

>>> s.run("table\_sample;")

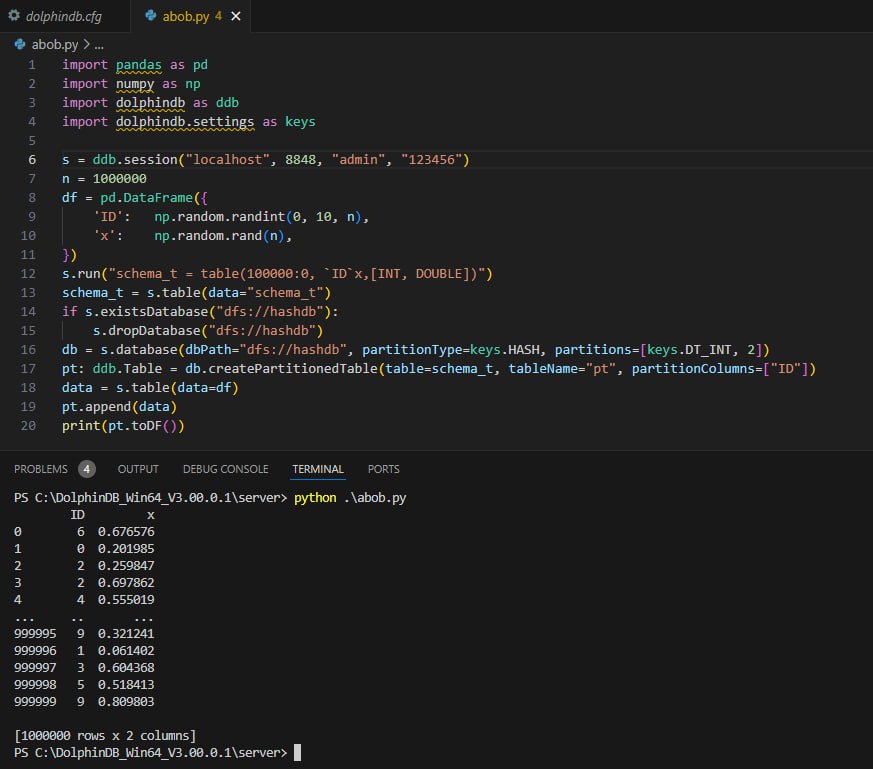
a b

0 1 a

1 2 b

2 3 c

Пример взаимодействия с БД через АПИ

  
Пример взаимодействия через SQL-скрипт

import dolphindb as ddb

s = ddb.session()

s.connect("localhost", 8848, "admin", "123456")

s.run("""

n=1000000

ID=rand(10, n)

x=rand(1.0, n)

t=table(ID, x)

db=database(directory="dfs://hashdb", partitionType=HASH, partitionScheme=[INT, 2])

pt = db.createPartitionedTable(t, `pt, `ID)

pt.append!(t);

""")

re = s.run("select count(x) from pt;")

print(re)

PS C:\DolphinDB\_Win64\_V3.00.0.1\server> python ./abob.py

count\_x

0 1000000

* 1. На каком языке/ах программирования написана СУБД?  
     C++
  2. Какие типы индексов поддерживаются в БД? Приведите пример создания индексов.  
     В DolphinDB поддерживаются следующие типы индексов:

**B-tree** — это самый распространённый тип индекса, который используется для поиска по одному или нескольким столбцам.

**Hash** — это тип индекса, который используется для поиска по одному столбцу.

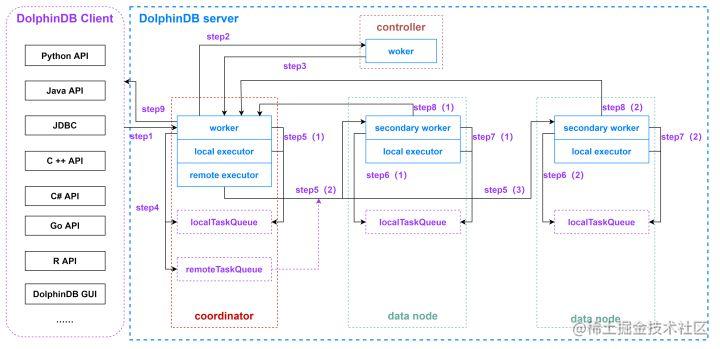
**Bitmap** — это тип индекса, который используется для поиска по нескольким столбцам, но только для логических значений.

Пример создания индекса:

CREATE INDEX CustomerID\_Index ON Customers (CustomerID)

Этот запрос создаст **B-tree** индекс по столбцу CustomerID в таблице Customers.

Чтобы создать **Hash** надо написать USING HASH (Username)

* 1. Как строится процесс выполнения запросов в вашей СУБД?  
       
     1. Клиент **DolphinDB** бинарно сериализует запросы и передаёт их узлу данных по протоколу TCP.
     2. После получения запроса от клиента узел данных выделяет рабочий поток для десериализации и анализа содержимого. Если содержимое оказывается SQL-запросом, то делается запрос к контроллеру для получения информации обо всех разделах, связанных с запросом.
     3. После получения запроса от узла данных контроллер выделяет рабочий поток для десериализации и анализа содержимого.
     4. После подготовки информации о разделах данных, участвующих в SQL-запросе, она сериализуется рабочим потоком и передаётся узлу данных по протоколу TCP.
     5. После получения информации, возвращённой контроллером, узел данных десериализует и анализирует её рабочим потоком.
     6. После получения промежуточных результатов вычислений, возвращённых удалёнными узлами, рабочий поток десериализует содержимое, а затем вычисляет окончательный результат и передаёт его клиенту через протокол TCP после сериализации
  2. Есть ли для вашей СУБД понятие «план запросов»? Если да, объясните, как работает данный этап.

Процесс выполнения запросов в **DolphinDB** строится следующим образом:

* + 1. **FROM**. Определяет таблицы, задействованные в запросе.
    2. **JOIN**. Выполняет операцию объединения на основе условия объединения.
    3. **WHERE**. Применяет условие фильтрации к объединённой таблице.
    4. **GROUP BY**. Группирует строки по указанным столбцам.
    5. **HAVING**. Фильтрует группы по условию.
    6. **SELECT**. Выбирает столбцы и агрегатные функции из каждой группы.
    7. **ORDER**. Сортирует строки по указанным столбцам.
    8. **LIMIT**. Пропускает несколько строк из отсортированного набора результатов.

*Распределенные запросы*:

* + 1. Система определяет все соответствующие разделы на основе условия where.
    2. Затем она переписывает запрос и отправляет его на узлы, где находятся соответствующие разделы.
    3. Наконец, система объединяет результаты из всех соответствующих разделов.
  1. Поддерживаются ли транзакции в вашей СУБД? Если да, то расскажите о нем. Если нет, то существует ли альтернатива?  
     Транзакции в **DolphinDB** поддерживаются.

Транзакция — это последовательность операций, которая выполняется как одна операция. Транзакции обеспечивают целостность данных и позволяют выполнять сложные операции.

* 1. Какие методы восстановления поддерживаются в вашей СУБД. Расскажите о них.  
     В DolphinDB поддерживаются следующие методы восстановления:
     1. **Резервное копирование** — это метод восстановления, который позволяет восстановить данные после сбоя.
     2. **Репликация** — это метод восстановления, который позволяет восстановить данные после отказа узла.

Резервное копирование в **DolphinDB** осуществляется с помощью команды BACKUP. Команда BACKUP создаёт резервную копию базы данных или таблицы.

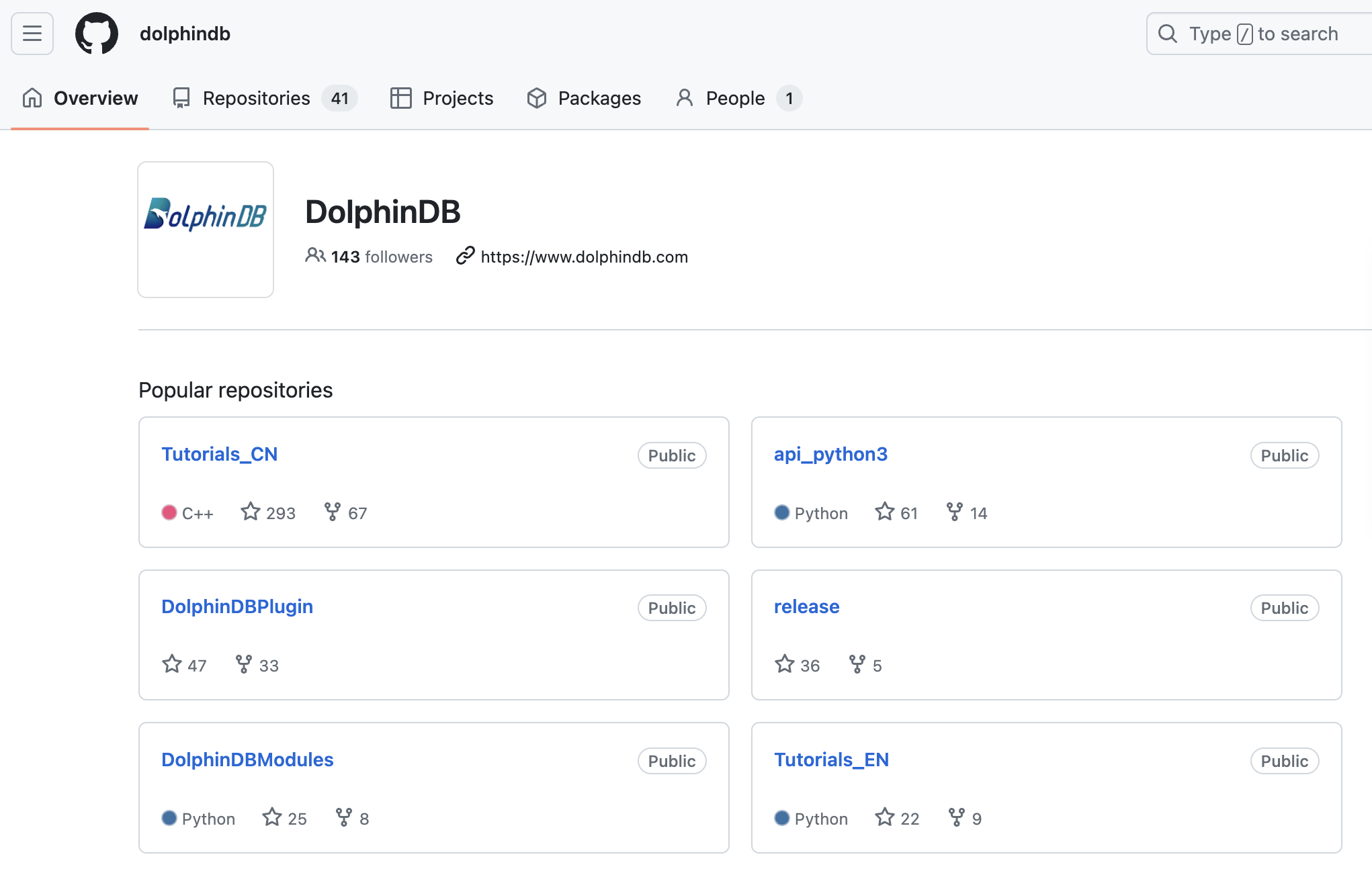
Репликация в **DolphinDB** осуществляется с помощью команды REPLICATE. Команда REPLICATE создаёт реплику базы данных или таблицы.

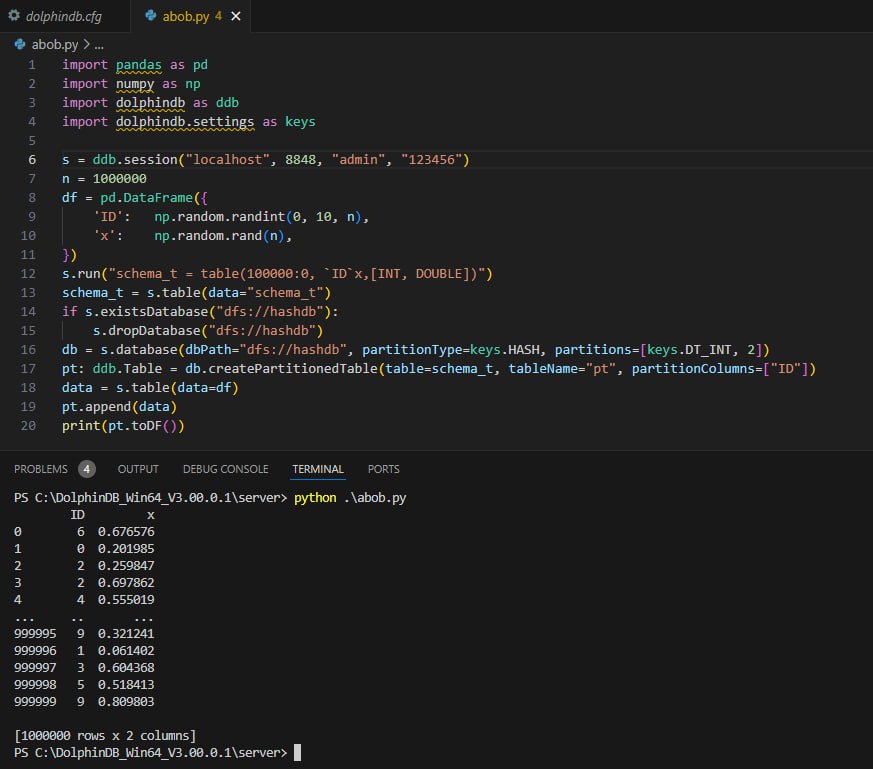
* 1. Расскажите про шардинг в вашей конкретной СУБД. Какие типы используются? Принцип работы.  
     В **DolphinDB** используется метод горизонтального партиционирования.
     1. **RANGE.** Разбиения определяются диапазонами, указанными любыми двумя смежными элементами вектора схемы разбиения. Начальное значение является включающим, а конечное - исключающим. Строка со значением столбца разбиения, попадающая в диапазон, принадлежит разбиению, определяемому этим диапазоном.  
        db=database("dfs://rangedb", RANGE, 0 5 10)   
        pt=db.createPartitionedTable(t, `pt, `ID)
     2. **VALUE.** Разбиение определяется значением какого-то заданного столбца.  
        db=database("dfs://valuedb", VALUE, 2000.01M..2016.12M)   
        pt = db.createPartitionedTable(t, `pt, `month)
     3. **HASH.** Данные разбиваются на шарды на основе хеш-функции, которая принимает входные данные и возвращает хеш-значение.   
        db=database("dfs://hashdb", HASH, [INT, 2])   
        pt = db.createPartitionedTable(t, `pt, `ID)
     4. **LIST.** Каждый элемент вектора схемы разбиения соответствует разделу**.**db=database("dfs://listdb", LIST, [`IBM`ORCL`MSFT, `GOOG`FB])   
        pt = db.createPartitionedTable(t, `pt, `ticker)
     5. **COMPO**. Может содержать 2 или 3 столбца Шардирования. Каждый столбец может иметь диапазон, значение, список или хэш-область.  
        dbDate = database(, VALUE, 2017.08.07..2017.08.11)   
        dbID = database(, RANGE, 0 50 100)   
        db = database("dfs://compodb", COMPO, [dbDate, dbID])   
        pt = db.createPartitionedTable(t, `pt, `date`ID)
  2. Возможно ли применить термины Data Mining, Data Warehousing и OLAP в вашей СУБД?

Термины Data Mining, Data Warehousing и OLAP можно применить к DolphinDB.

* 1. Какие методы защиты поддерживаются вашей СУБД? Шифрование трафика, модели авторизации и т.п.
     1. **Шифрование кода**. В DolphinDB есть функция шифрования, которая позволяет шифровать скрипты и пользовательские функции. Это предотвращает несанкционированный доступ к исходному коду.
     2. **Контроль доступа к сценариям**. В DolphinDB есть механизмы контроля доступа, которые позволяют ограничить доступ к конкретным сценариям или функциям. Можно определить роли пользователей и разрешения, чтобы доступ к алгоритмам имели только авторизованные пользователи или администраторы.
     3. **Удаленное выполнение**. DolphinDB поддерживает удалённое выполнение сценариев, когда можно отправлять данные на сервер и выполнять вычисления там без раскрытия фактического кода на стороне клиента.
     4. **Безопасное общение**. Необходимо обеспечить безопасное общение между клиентом и сервером DolphinDB с использованием протоколов шифрования, таких как SSL/TLS. Это защитит данные и алгоритмы во время передачи.
     5. **Регулярные обновления и патчи**. Важно следить за последними обновлениями и патчами DolphinDB. Это поможет защитить систему от известных проблем безопасности и уязвимостей.
     6. **Физическая безопасность**. Необходимо ограничить физический доступ к серверу DolphinDB только авторизованным лицам. Физическая безопасность — важный аспект защиты конфиденциальных данных и кода.
  2. Какие сообщества развивают данную СУБД? Кто в проекте имеет права на коммит и создание дистрибутива версий? Расскажите об этих людей и/или компаниях.  
     Сообщество разработчиков DolphinDB включает в себя сотрудников компании Dolphindb Inc., а также независимых разработчиков.

Права на коммит и создание дистрибутива версий в проекте DolphinDB имеют сотрудники компании Dolphindb Inc.



* 1. Создайте свои собственные данные для демонстрации работы СУБД.   
     Создание дынных из DataFrame Pandas  
     
  2. Как продолжить самостоятельное изучение языка запросов с помощью демобазы. Если демобазы нет, то создайте ее.

Код создания тестовой базы данных

sql

login("admin", "123456")

sql

// Define the database path

dbPath = "dfs://ExampleDB"

// Create database

db = database(dbPath, VALUE, 2019.01.01..2021.12.31)

// Define a table schema

t = table(100:0, iddatesymbolprice`volume, [INT, DATE, SYMBOL, DOUBLE, INT])

// Persist this table in the database. 'trades' is the table name.

db.createPartitionedTable(t, trades, date)

Код наполнения базы данных данными

sql

// Generate sample data

dates = take(2019.01.01 + 0..1000, 10)

symbols = take(AAPLGOOGAMZNMSFT, 10)

prices = take(100.0 + rand(50, 10), 10)

volumes = take(100 \* rand(100, 10), 10)

// Insert data into the table

tableInsert(db.loadTable("trades"), iddatesymbolprice`volume, 1..10, dates, symbols, prices, volumes)

* 1. Где найти документацию и пройти обучение  
     <https://docs.dolphindb.com/en/index.html>  
     
  2. Как быть в курсе происходящего  
     <https://dolphindb.com/news>  
     