(GridDBと一緒に)
Pythonの標準インタフェースDB-API(2.0)を使ってみましょう
～JPype DBAPI2の利用方法とApache Arrowによる高速化～

TOSHIBA

東芝デジタルソリューションズ株式会社　GridDBコミュニティ版担当
野々村 克彦

2025.02.21

© 2025 Toshiba Digital Solutions Corporation

プロフィール

Katsuhiko Nonomura
knonomura

名前：野々村 克彦（ののむら かつひこ）
所属：
- (株)東芝　デジタルイノベーションテクノロジーセンター
  技術開発室 ソフトウェア開発部
- 東芝デジタルソリューションズ株式会社
  ソフトウェアシステム技術開発センター　ソフトウェア開発部　兼務

経歴：
１９９３年　東芝　研究開発センター入所
　　　　　　ルールベースの不良解析システム開発に従事
１９９８年～　XMLデータベース開発に従事
２０１２年～　IoT向けデータベースGridDB開発に従事
２０１５年～　GridDBのオープンソースPJ開始
現在　GridDB保守、製品版開発（PythonAPIなど）
　　　GridDB OSS版開発、OSS活動

© 2025 Toshiba Digital Solutions Corporation

Contents

01 はじめに

02 DB-API(2.0)とJPype DBAPI2

03 インストール方法

04 操作方法（基本編）

05 操作方法（応用編）

06 結果取得の高速化

07 まとめ

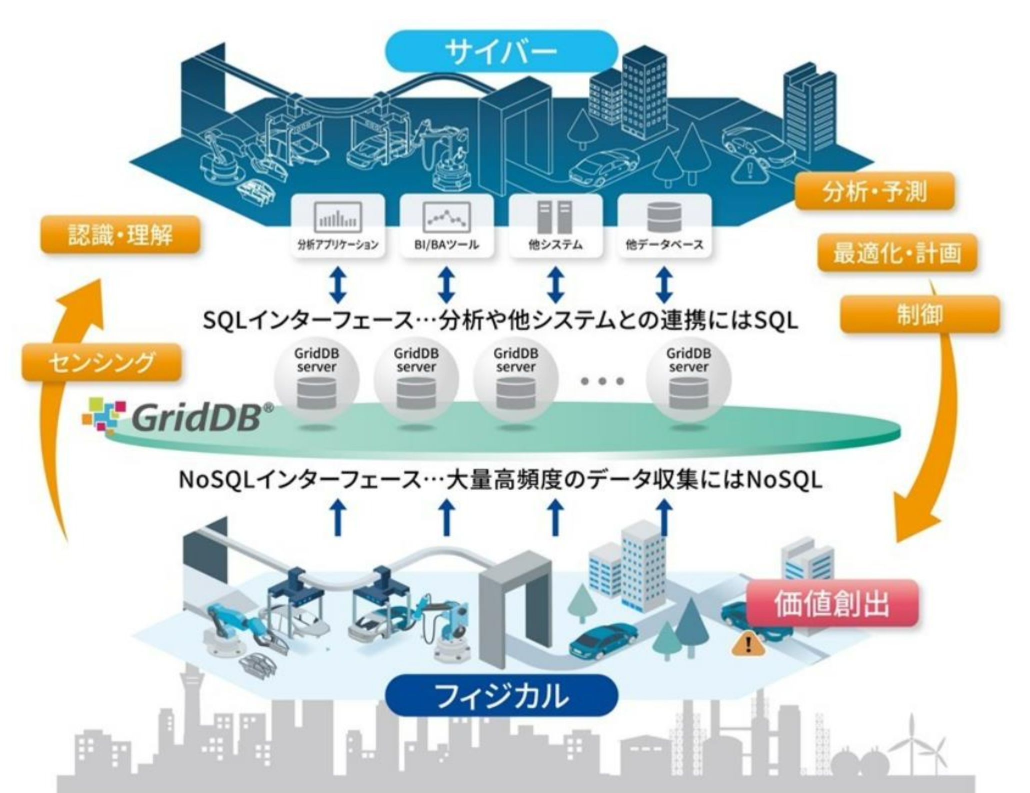
© 2025 Toshiba Digital Solutions Corporation

01

はじめに

（今回のプレゼンの説明で利用するデータベース）GridDBについて

- GridDBとは、東芝デジタルソリューションズが開発した日本発のNoSQL型のDBMSです。
- ビッグデータやIoTシステム向けに特化して作られたDBMSです。
- **NoSQL（キーバリュー型）インターフェース**だけではなく、**SQLインターフェース**を提供します。（デュアルインターフェース）



ラインアップ

GridDB
Enterprise Edition

高頻度・大量に発生する時系列データの蓄積とリアルタイムな活用をスムーズに実現し、ビジネスを大きく成長させるために最適化された次世代のデータベース

GridDB Cloud

高頻度・大量に発生する時系列データの蓄積とリアルタイムな活用をスムーズに実現するクラウドデータベースサービス

2016年 GitHub上にオープンソース化
最新版はV5.7
https://github.com/griddb/griddb

© 2025 Toshiba Digital Solutions Corporation

GridDB Cloudを無料で使ってみませんか？

griddb cloud 検索

GridDB Cloud無料プラン

申し込みフォームがより簡単になりました。必要なのは「お名前」と「メールアドレス」のみですので、ぜひお試しください。

https://www.global.toshiba/jp/products-solutions/ai-iot/griddb/product/griddb-cloud.html

TOSHIBA 東芝デジタルソリューションズ株式会社

サイトマップ お問い合わせ

ホーム | ソリューション・サービス | 製品 | イベント情報 | ニュース | 企業情報 | 事例紹介

HOME > GridDB Cloud > 無料プラン申し込み

GridDB Cloud 無料プラン申し込みフォーム

GridDBクラウドサービス無料プランはこちらのページからお申込みください。

以下のフォームに必要事項をご記入の上、「確認する」ボタンをクリックして内容を確認し、「送信」ボタンをクリックしてください。

必須は必須入力項目です。

お名前 必須	姓 記入例：東芝　名 記入例：太郎
Email アドレス 必須	半角でご入力ください
Email アドレス（確認）必須	確認のため、再度ご入力ください

https://form.ict-toshiba.jp/download_form_griddb_cloud_freeplan

© 2025 Toshiba Digital Solutions Corporation

02

DB-API(2.0)とJPype DBAPI2

© 2025 Toshiba Digital Solutions Corporation

各種開発言語のSQLインタフェース

	Java言語	Python言語	Go言語
DB操作仕様・API	JDBC	**DBAPI2**	database/sql
名称	Java DataBase Connectivity	PEP 249 – Python Database API Specification v2.0	database/sql package
URL	https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/index.html	https://peps.python.org/pep-0249/	https://pkg.go.dev/database/sql
主要クラス	• Connection • Statement • PreparedStatement • ResultSet • DatabaseMetaData • ParameterMetaData • ResultSetMetaData	• Connection • Cursor 非常にシンプル	• Conn • DB • Stmt • Rows • Tx

© 2025 Toshiba Digital Solutions Corporation

DB-API(2.0)とは

- PEP 249 – Python Database API Specification v2.0 (https://peps.python.org/pep-0249/)
  - Python言語のDBアクセス用の**API仕様**
  - MySQL、PostgreSQL、Oracle、SQL Server、SQLiteなど主なRDBでサポートされている。
  - 非常にシンプル
- 主なAPI一覧

	クラス	メソッド	機能概要
接続		connect(*parameters*…)	接続
SQL実行	Cursor	execute(*operation* [, *parameters*])	SQL文の実行
	Cursor	executemany(*operation*, *seq_of_parameters*)	N個のSQL文の実行
	Cursor	callproc(*procname* [, *parameters*])	プロシージャコール
フェッチ	Cursor	fetchone()	1件フェッチ
	Cursor	fetchmany([*size=cursor.arraysize*])	N件フェッチ
	Cursor	fetchall()	全件フェッチ

© 2025 Toshiba Digital Solutions Corporation

JPype DBAPI2とは

- JPype内のモジュールの１つ。JDBCドライバ上にDB-API(2.0)仕様による操作（実装）を提供する**ソフト(OSS)**　https://jpype.readthedocs.io/en/latest/dbapi2.html

  ※ **JPype**：https://jpype.readthedocs.io/en/latest/index.html

  Python からJNI(Java Native Interface)経由で Java アクセスを提供するソフト(OSS)

- DB-API(2.0)仕様による操作以外に以下に対応している。
  - Python/Java間のデータ型変換のカスタマイズ
  - JDBCドライバのオブジェクト取得⇒JDBCによる操作（メタデータ参照など）
    - Connectionオブジェクト、ResultSetオブジェクト

© 2025 Toshiba Digital Solutions Corporation

補足：JDBCベースのPython DBAPI2実装について

- JayDeBeAPI (https://github.com/baztian/jaydebeapi)
  - 2020年ごろまで、よく使われていた。
  - 2020年6月以降の活動はほとんど無い。
  - JPype 0.6.3（かなり古い版）で動く。新しい版に移行しなかった。
- JPype DBAPI2 (https://github.com/jpype-project/jpype)
  - JPypeのモジュールの１つ。JPype1.1にて対応(2020年10月)。JPype最新は1.5。

  ※ JayDeBeAPIが古いJPypeに依存したままであり、JPypeユーザが古い版に固執してしまったため、JPype開発者がDBAPI2に対応した経緯あり。JayDeBeAPIの問題点も指摘。

  https://github.com/jpype-project/jpype/pull/744

© 2025 Toshiba Digital Solutions Corporation

03

インストール

- GridDB
- JPype

© 2025 Toshiba Digital Solutions Corporation

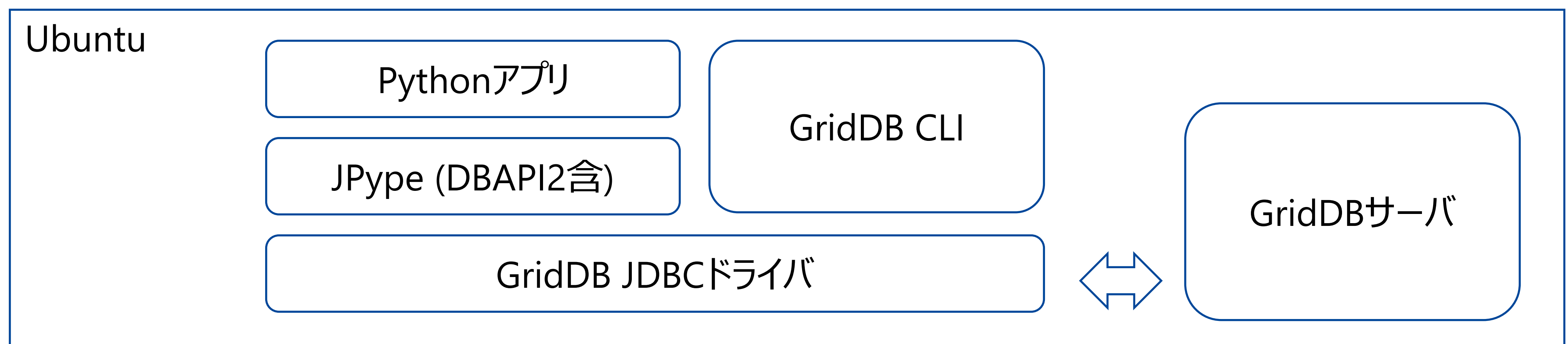
動作環境の例

<動作環境>

- Windows 11上にWSL2 Ubuntu 22.04。Python3、Java 11(デフォルト)インストール済
- メモリは16GB（ブラウザ、Zoom等のアプリを動かさないのであれば8GBも可）

<前提条件>

- 同一マシンに全ソフトウェアをインストール。ローカル実行
- GridDBのクラスタ名はmyCluster(デフォルト)
- GridDB管理者の名前はadmin、パスワードはadmin



※GridDBサーバ：https://github.com/griddb/griddb
※GridDB JDBCドライバ： https://github.com/griddb/jdbc
※GridDB CLI：https://github.com/griddb/cli

© 2025 Toshiba Digital Solutions Corporation

① GridDBのインストール＆起動の手順

【インストール】

1. GridDBサーバのインストール

```
$ wget https://github.com/griddb/griddb/releases/download/v5.7.0/griddb_5.7.0_amd64.deb
$ sudo dpkg -i griddb_5.7.0_amd64.deb
```

2. GridDB CLI（コマンドライン・インタフェース）のインストール

```
$ wget https://github.com/griddb/cli/releases/download/v5.7.0/griddb-ce-cli_5.7.0_amd64.deb
$ sudo dpkg -i griddb-ce-cli_5.7.0_amd64.deb
```

【起動】

3. GridDBのサービス起動

```
$ sudo systemctl start gridstore
```

4. CLI起動

```
$ sudo su - gsadm
$ gs_sh
```

[SQLの例]

```
# テーブル作成
> create table t1 (c0 long, c1 long);
# データ登録
> insert into t1 values(1, 2);
# 検索
> select * from t1;
> get
```

※GridDBサービスの停止

```
$ systemctl stop gridstore
```

※ サービス機能(systemctl)が利用できない環境の場合は、gs_startnode等の運用コマンドをご利用ください。

© 2025 Toshiba Digital Solutions Corporation

① GridDBのインストール＆起動の手順

【インストール】

1. GridDBサーバのインストール

```
$ wget https://github.com/griddb/griddb/releases/download/v5.7.0/griddb_5.7.0_amd64.deb
$ sudo dpkg -i griddb_5.7.0_amd64.deb
```

2. GridDB CLI（コマンドライン・インタフェース）のインストール

```
$ wget https://github.com/griddb/cli/releases/download/v5.7.0/griddb-ce-cli_5.7.0_amd64.deb
$ sudo dpkg -i griddb-ce-cli_5.7.0_amd64.deb
```

【起動】

3. GridDBのサービス起動

```
$ sudo systemctl start gridstore
```

4. CLI起動

```
$ sudo su - gsadm
$ gs_sh
```

[SQLの例]

```
# テーブル作成
> create table t1 (c0 long, c1 long);
# データ登録
> insert into t1 values(1, 2);
# 検索
> select * from t1;
> get
```

わずかなステップだけで
CLIによるSQLなどの操作が開始できます。

※GridDBサービスの停止

```
$ systemctl stop gridstore
```

※ サービス機能(systemctl)が利用できない環境の場合は、gs_startnode等の運用コマンドをご利用ください。

© 2025 Toshiba Digital Solutions Corporation

② JPypeのインストール

【インストール】

1. JPypeのインストール

$ pip install JPype1==1.5.0

$ mvn dependency:get -Dartifact=com.github.griddb:gridstore-jdbc:5.7.0 -Ddest=./gridstore-jdbc.jar

⇒ カレントフォルダにgridstore-jdbc.jarファイルが配置されます。

© 2025 Toshiba Digital Solutions Corporation

04

操作方法（基本編）

- 事前処理
- 接続
- SQL実行
- 検索結果の取得

© 2025 Toshiba Digital Solutions Corporation

①事前処理

● 以下のインポートとJVMの起動が事前に必要です。

```
import jpype

import jpype.dbapi2

jpype.startJVM(classpath=['./gridstore-jdbc.jar'])
```

© 2025 Toshiba Digital Solutions Corporation

②接続

- connect()メソッドの入力値は各データベースごとに異なります。

  以下を指定してください。 【GridDB仕様】

  - url: 接続に使うURL。GridDB JDBCドライバと同じ
    - 接続先サーバのIDアドレス、SQLインタフェースのポートNo、GridDBクラスタ名などを指定
  - driver: "com.toshiba.mwcloud.gs.sql.Driver"
  - driver_args: GridDBサーバに接続するためのGridDBユーザ名とパスワード

- 接続の例：

```
url = "jdbc:gs://127.0.0.1:20001/myCluster"

conn = jpype.dbapi2.connect(url, driver="com.toshiba.mwcloud.gs.sql.Driver",

      driver_args={"user":"admin", "password":"admin"})
```

© 2025 Toshiba Digital Solutions Corporation

補足：SQL概要

- SQL（Structured Query Language）：データベース操作言語

分類	主な操作	SQLの例	意味
データ定義言語（DDL）	テーブル生成	CREATE TABLE Sample (id integer PRIMARY KEY, value string);	数値のidカラムと文字列のvalueカラムをもつ、Sampleテーブルの作成
データ操作言語（DML）	データ挿入	INSERT INTO Sample values (0, 'test0’);	Sampleテーブルへのロウデータ(0, ‘test0’)の挿入
	検索	SELECT * FROM Sample WHERE id > 0;	Sampleテーブルに対し、id値が0より大きいデータの検索
データ制御言語（DCL）	アクセス権限の付与	GRANT ALL ON db1 TO user1;	データベースdb1のユーザuser1に全権限を付与

Sampleテーブル

idカラム	valueカラム
0	‘test0’

ロウ

© 2025 Toshiba Digital Solutions Corporation

③SQLの実行 (1/2)

(A) execute() SQL 1個の実行

- SQLの実行

curs = conn.cursor()

curs.execute(“CREATE TABLE Sample (id integer PRIMARY KEY, value string);”) テーブル生成

curs.execute(“INSERT INTO Sample values (0, ‘test0’);”) データ挿入

curs.execute(“INSERT INTO Sample values (1, ‘test1’);”) データ挿入

curs.execute(“SELECT * from Sample where ID > 0;”) 検索

print(curs.fetchall())

print出力

[[1, ’test1’]]

- プレースホルダ付きSQLの実行

curs = conn.cursor()

curs.execute(“INSERT INTO Sample values (?, ?);“, (0, ‘test0’))

※ 利用可能なプレースホルダは「?」記号に限定
【JPype DBAPI2仕様】

© 2025 Toshiba Digital Solutions Corporation

③SQLの実行 (2/2)

(B) executemany() SQL N個の実行

curs = conn.cursor()

data = [(0, ‘test0’), (1, ‘test1’), (2, ‘test2’) , (3, ‘test3’)]

curs.executemany(“INSERT INTO Sample values (?, ?)“, data)

(C) プロシージャコール･･･【GridDB仕様】 GridDBでは未サポート

curs = conn.cursor()

curs.callproc(“lower”, (“FOO”,))

※ lower : 入力文字列を小文字に変換する関数

© 2025 Toshiba Digital Solutions Corporation

④検索結果の取得

Sampleテーブル

idカラム	valueカラム
0	‘test0’
1	‘test1’
2	‘test2’
3	‘test3’

(A) fetchone() 1件取得

```
curs.execute("SELECT * from Sample where id > 0")
print(curs.fetchone())
print(curs.fetchone())
print(curs.fetchone())
```

print出力

```
[[1, ’test1’]]
[[2, ’test2’]]
[[3, ’test3’]]
```

(B) fetchmany() N件取得

```
curs.execute("SELECT * from Sample where id > 0")
print(curs.fetchmany(2))
print(curs.fetchmany(2))
```

print出力

```
[[1, ’test1’], [2, ‘test2’]]
[[3, ’test3’]]
```

(C) fetchall() 全件取得

```
curs.execute("SELECT * from Sample where id > 0")
print(curs.fetchall())
```

print出力

```
[[1, ’test1’], [2, ‘test2’], [3, ’test3’]]
```

© 2025 Toshiba Digital Solutions Corporation

04

操作方法（応用編）

- TIMESTAMP型のナノ秒精度
- Int/float型
- Blob型

© 2025 Toshiba Digital Solutions Corporation

課題１：検索結果の取得時のデータ型について

	デフォルト状態での制限事項	理由
TIMESTAMP型	ナノ秒精度での設定はできるが、ナノ秒精度で取得ができない。【JPype DBAPI2仕様】	デフォルトではPython datetime型にマッピングされるが、datetimeはマイクロ秒精度であるため。
数値型	取得結果がJavaクラスをラッピングしたデータ型になる。 例：INTEGER型⇒JInt型【JPype DBAPI2仕様】	デフォルトではJavaクラスをラッピングしたデータ型にマッピングされるため
BLOB型	取得はできるが、設定ができない。【GridDB仕様】	デフォルトでは、GridDB未サポートのsetBytes()メソッドを使うため

- Jpype DBAPI2のカスタマイズ機能（Adapters/Convertersを定義or変更）で、データ型変換の動作を変更できる。
  - Adapters
    - パラメータを設定するときに、Python型からJava型に変換するために使用されます。
  - Converters
    - 結果が生成されるとコンバータを使用して、Java型をPython型に変換し直すことができます。

⇒ 今回、**デフォルトのAdapters/Convetersを変更する方法**でカスタマイズする

© 2025 Toshiba Digital Solutions Corporation

ご参考：デフォルトのAdapters/Converters

```
>> conn._adapters
{}            ←空

>> conn._converters
{<java class ‘java.lang.String’>: <class ‘str’>, ←pythonのstr型に変換
<java class ‘java.sql.Date’>: <function _asPython at 0x…>,
<java class ‘java.sql.Time’>: <function _asPython at 0x…>,
<java class ‘java.sql.Timestamp’>: <function _asPython at 0x…>
                                        ←JDBCTypeのPyTypesに従い、datetime型に変換
<java class ‘java.math.BigDecimal’>: <function _asPython at 0x…>,
<java class ‘byte[]’>: <class ‘bytes’>,
<class ‘NoneType’>: <function _nop at 0x…>}    ←変換しないでそのまま
```

(A) TIMESTAMP型 ナノ秒の取得方法

デフォルトConvertersの変更

```
curs.execute(“CREATE TABLE IF NOT EXISTS SampleNano ( id integer PRIMARY KEY, t
TIMESTAMP(9) )”) ･･･【GridDB仕様】TIMESTAMP型でナノ秒精度を使う場合
curs.execute("INSERT INTO SampleNano values (1, TIMESTAMP_NS('2023-05-
18T12:00:00.123456789Z'))")
curs.execute("SELECT * from SampleNano")

import jpype.imports
import java.sql.Timestamp
conn._converters[java.sql.Timestamp] = jpype.dbapi2._nop

row = curs.fetchone()
print(row)
print(*row)
```

変更前の処理：
Java Timestamp → Python datetime
変更後の処理：
Java Timestamp → 変換しない

変更前のprint出力：
```
[1, datetime.datetime(2023,5,18,12,00,00,123456)]
1 2023-05-18 12:00:00.123456
```
変更後のprint出力：
```
[1, <java.object ‘java.sql.Timestmp’>]
1 2023-05-18 12:00:00.123456789
```

© 2025 Toshiba Digital Solutions Corporation

(B) Pythonデータ型(int/float)での取得方法

デフォルトConvertersの変更

```
curs.execute("CREATE TABLE SampleNum ( id integer PRIMARY KEY, v1 LONG, v2 SHORT, v3 BYTE, v4
FLOAT, v5 DOUBLE )")
curs.execute("INSERT INTO SampleNum values (0, 1, 2, 3, 4, 5)")
curs.execute("SELECT * from SampleNum")

conn._converters[jpype.JByte] = int
conn._converters[jpype.JShort] = int
conn._converters[jpype.JInt] = int
conn._converters[jpype.JLong] = int
conn._converters[jpype.JFloat] = float
conn._converters[jpype.JDouble] = float

row = curs.fetchone()
print(row)
print([type(v) for v in row])
```

変更前の処理：
JInt → 変換なし
変更後の処理：
JInt → int

修正前のprint出力：
[0, 1, 2, 3, 4.0, 5.0]
<java class ‘JShort’> <java class ‘JShort’> <java class ‘JInt’> <java class ‘JLong’> <java class ‘JDouble’> <java class ‘JDouble’>
修正後のprint出力：
[0, 1, 2, 3, 4.0, 5.0]
<class ‘nt’> <class ‘int’> <class ‘int’> <class ‘int’> <class ‘float’> <class ‘float’>

(X) BLOB型の値設定のカスタマイズ方法

デフォルトAdapters/settersの変更

```
import jpype.imports
from javax.sql.rowset.serial import SerialBlob

curs.execute("CREATE TABLE SampleBlob ( id integer PRIMARY KEY, v1 BLOB )”)

conn._adapters[bytes] = SerialBlob
jpype.dbapi2._default_setters[SerialBlob] = jpype.dbapi2.BLOB

b = "abcde".encode()
curs.execute("INSERT INTO SampleBlob values (?, ?)", (1, b))
b = "xyzvw".encode()
curs.execute("INSERT INTO SampleBlob values (?, ?)", (2, b))

curs.execute("SELECT * from SampleBlob")
print(curs.fetchall())
```

変更前の処理：
bytes → setBytes()利用
変更後の処理：
bytes → SerialBlob → setBlob()利用

変更前のprint出力：
データ挿入時にエラー
変更後のprint出力：
[[1, b’abcde’]]
[[2, b’xyzvw’]]

© 2025 Toshiba Digital Solutions Corporation

06

結果取得の高速化

- Apache Arrowによる高速化

課題２：検索結果の大量ヒット時の取得処理について

- 検索結果が大量ヒットの場合、取得に時間を要する
  - JNI(Java Native Interface)経由のJDBCドライバのメソッドコールのコストが小さくないため。

```
result = curs.fetchall()
※ リストで返す
```

```
while (rs.next()):
    val1 = rs.getXXX(1)
    val2 = rs.getYYY(2)
    …
```

⇒ Apache Arrowの利用

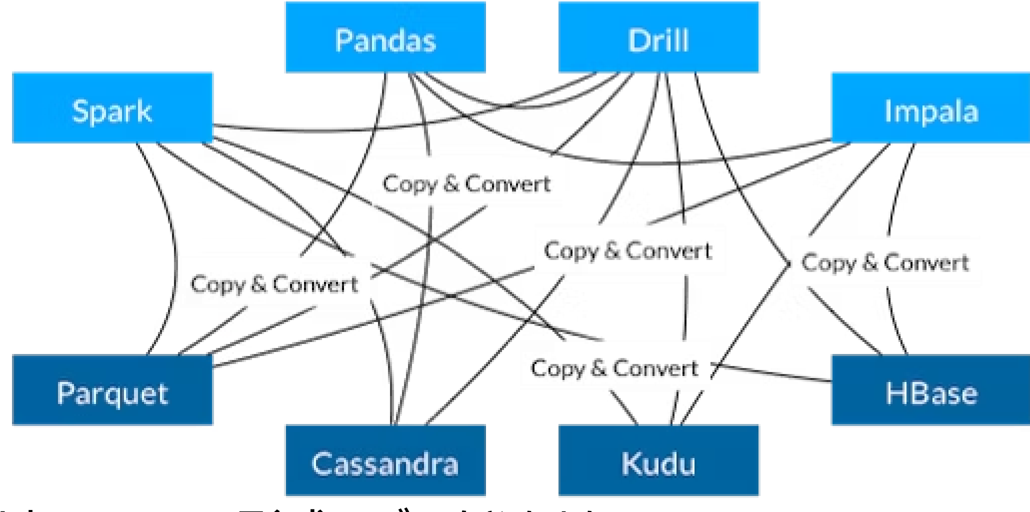
Apache Arrowとは

- Apacheプロジェクトの１つ
  - https://arrow.apache.org/
  - https://github.com/apache/arrow
  - Apacheライセンス
- 効率的なデータ交換のために設計されたデータフォーマット、S/W(OSS)
  - インメモリのカラム指向データフォーマット
- 向いている用途
  - 大量データの交換、メモリー上での大量データの分析処理
  - 利用例：Apache Arrowフォーマットを使うことにより、データ分析エンジンApache Sparkを約30倍高速化
    https://arrow.apache.org/blog/2017/07/26/spark-arrow/
- 2016年にファーストリリース、2020/7にV1.0リリース。最新はV19
- 様々な開発言語に対応
  - C, C++, C#, Go, Java, JavaScript, Julia, MATLAB, Python, R, Ruby, and Rust.
- Apache Parquet（ファイルフォーマット）やpandasと密な連携がなされている
  - pandasの原作者がApache Arrowを開発

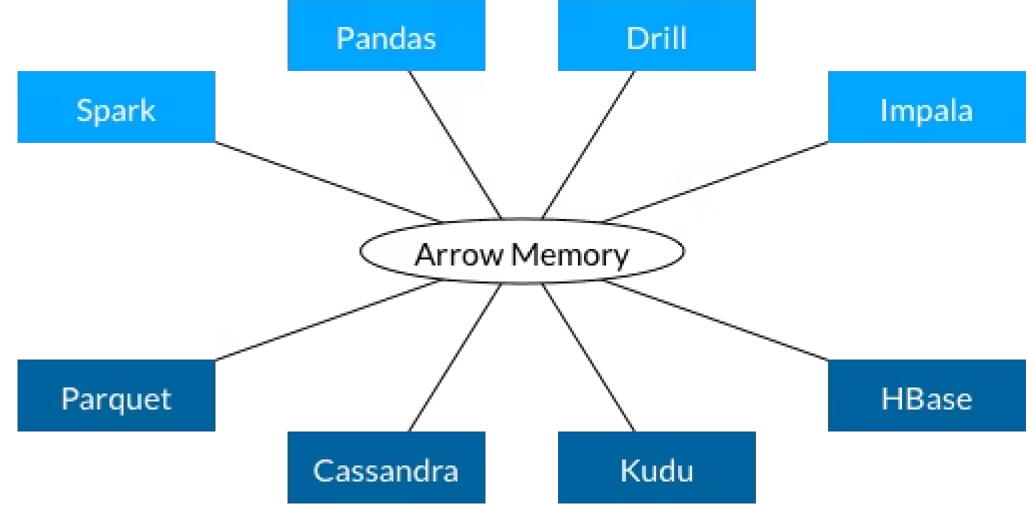
© 2025 Toshiba Digital Solutions Corporation

イメージ図

Arrowを利用しない場合：各ソフト、システム間で個々にデータのコピー＆変換



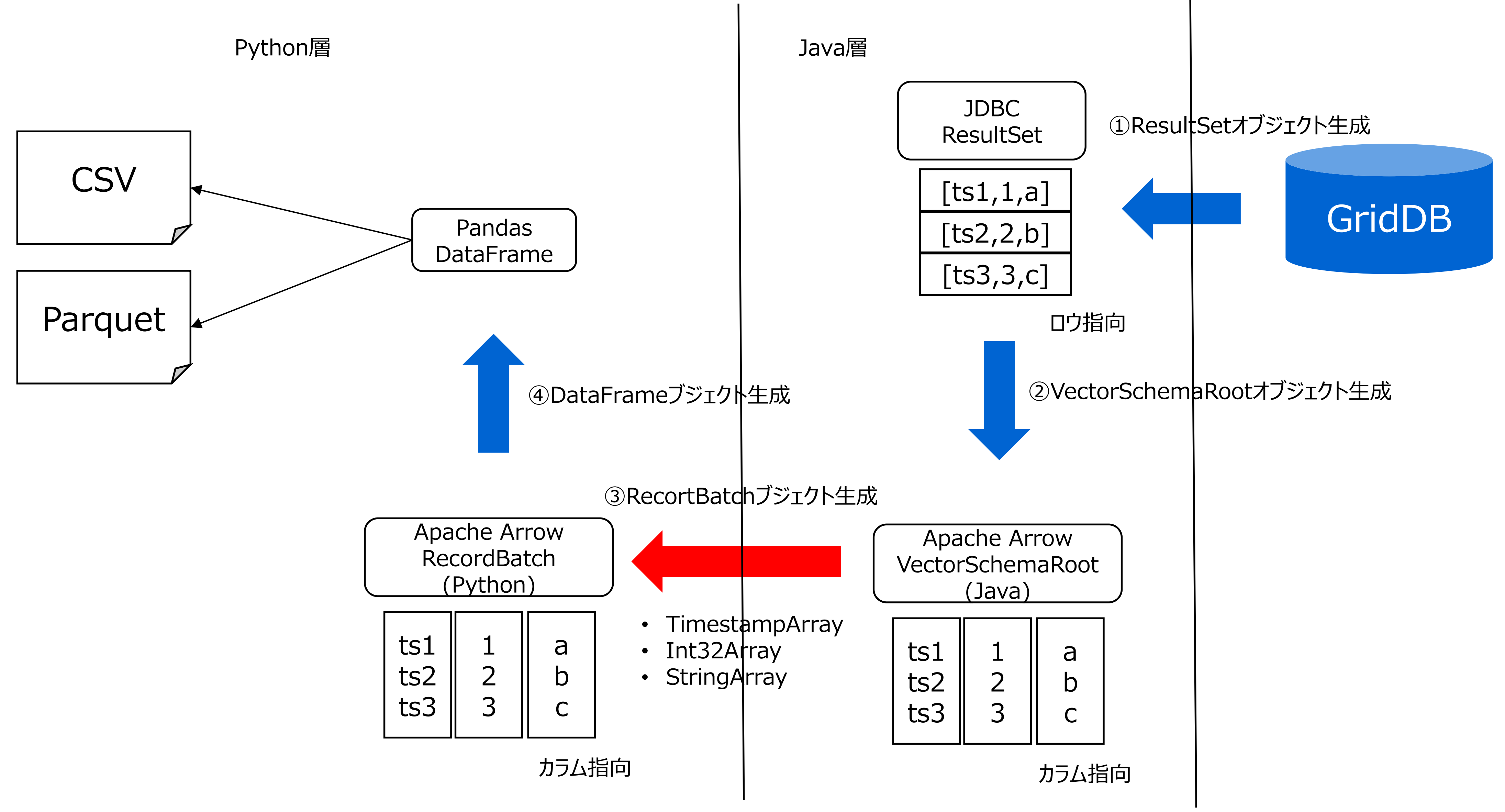
Arrowを利用した場合：メモリ上のArrow形式でデータやりとり



※ 図の出典：https://arrow.apache.org/overview/

© 2025 Toshiba Digital Solutions Corporation

検索結果の取得時の活用イメージ



※ RecordBatch(Python), VectorSchemaRoot(Java)：スキーマ情報とロウ配列情報を含むデータ構造

© 2025 Toshiba Digital Solutions Corporation

Apache Arrowのインストール

【インストール】

1. Apache Arrowのインストール・設定

1-1. Apache ArrowのPython層の処理用(pyarrow）

```
$ pip install pyarrow==16.0.0
```

1-2. Apache ArrowのJava層の処理用(Arrow JDBC Adapter)

```
$ wget https://github.com/griddb/jdbc/tree/master/sample/ja/python/dbapi2/jpype/pom.xml
$ mvn assembly:single
```

⇒ targetフォルダにgridstore-arrow-jdbc-0.1-jar-with-dependencies.jarファイルが生成されます。

```
$ export _JAVA_OPTIONS="--add-opens=java.base/java.nio=ALL-UNNAMED"
```

© 2025 Toshiba Digital Solutions Corporation

コード例

```
。。。
import jpype.imports
jpype.startJVM(classpath=["./target/gridstore-arrow-jdbc-0.1-jar-with-dependencies.jar"])

import pyarrow.jvm
from org.apache.arrow.adapter.jdbc import JdbcToArrowConfigBuilder, JdbcToArrow
from org.apache.arrow.memory import RootAllocator

ra = RootAllocator(sys.maxsize)
config_builder = JdbcToArrowConfigBuilder()
config_builder.setAllocator(ra)
config_builder.setTargetBatchSize(-1) # 全件取得時は-1、分割取得時は1回に取得する件数を設定
pyarrow_jdbc_config = config_builder.build()
```

```
curs.execute("SELECT *;")
```

```
result_set = curs.resultSet # JDBCのResultSetオブジェクトを取得
it = JdbcToArrow.sqlToArrowVectorIterator(result_set, pyarrow_jdbc_config)
while it.hasNext():
    root = it.next() # Java層でBatchSize分のデータを含むVectorSchemaRootオブジェクトを生成
    if root.getRowCount() == 0:
        break
    rb = pyarrow.jvm.record_batch(root) # Python層でRecordBatchオブジェクトを生成
    df = rb.to_pandas() # Pandas DataFrameオブジェクトへ変換
```

© 2025 Toshiba Digital Solutions Corporation

Fast JDBC access in Python using pyarrow.jvm (2020 edition)
(https://uwekorn.com/2020/12/30/fast-jdbc-revisited.html)
と同様の方法で測定。類似の結果が得られた。

測定条件：

- ニューヨーク市の黄色タクシーでの旅行記録データ（オープンデータ）を利用。
  名称：2016 Yellow Taxi Trip Data
  所有者：NYC OpenData
  提供元：Taxi and Limousine Commission (TLC)
  URL：https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/uacg-pexx/about_data
- SQLは「SELECT * FROM テーブル名 LIMIT <N>」を利用。
- SQL実行後のJDBC ResultSetオブジェクトから結果取得、Pandas Dataframeオブジェクト取得までを計測。

N	(A) JPype DBAPI2による結果取得時間（秒）	(B) Apache Arrowによる結果取得時間（秒）	(A)/(B)
1万	1.35	0.50	2.70
10万	14.86	0.81	18.12
100万	289.47	3.99	72.55

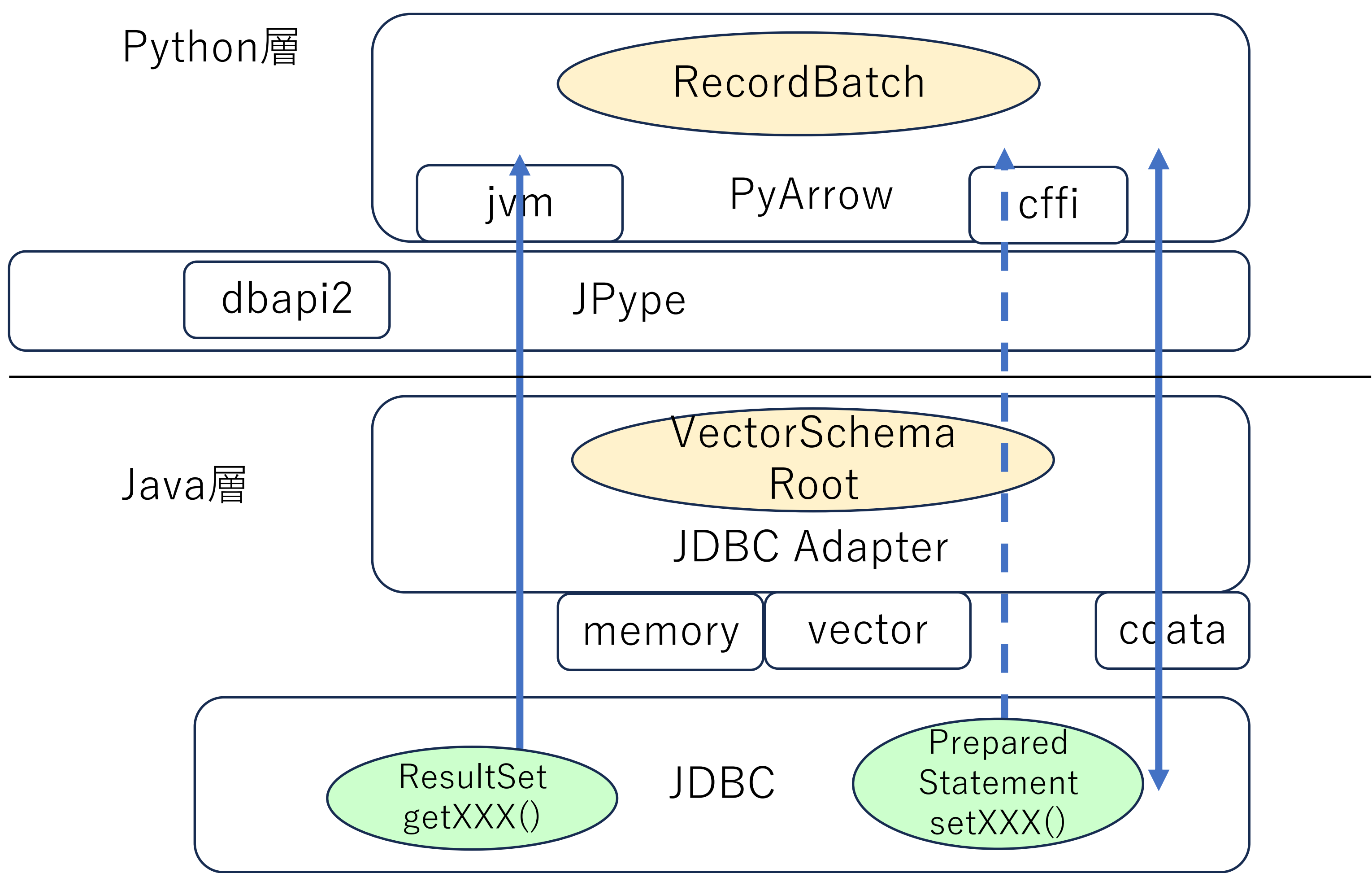
N（件数）が大きいほど差が出る

© 2025 Toshiba Digital Solutions Corporation

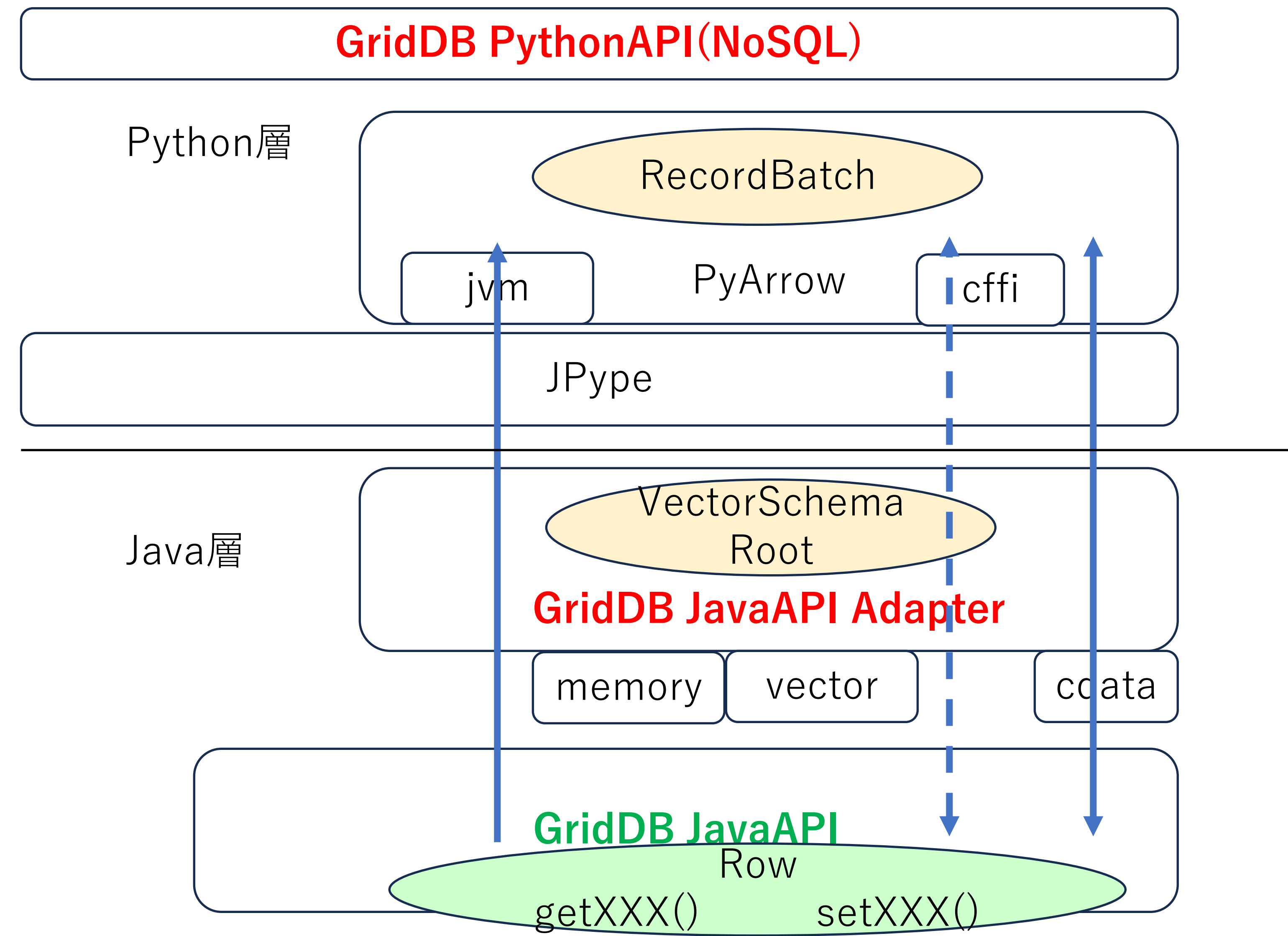
補足：Apache Arrow＋JDBCドライバのS/W構成

GitHub上のソース
https://github.com/apache/arrow

＜フォルダ構成＞
java/
c/main/java/…/
Data.java
adapter/
jdbc/
memory/
vector/
python/
pyarrow/
cffi.py
jvm.py



補足：Apache Arrow＋GridDB JavaAPIによるGridDB PythonAPI 構想



まとめの前に

本プレゼン内容に相当するドキュメント等を

GridDB CE V5.7のJDBCドライバ (https://github.com/griddb/jdbc)の

利用例として公開しております。よろしければ、ご参照ください。

- PythonAPI(SQL)ガイド
- サンプルコード

https://github.com/griddb/jdbc/tree/master/sample/ja/python/dbapi2/jpype

© 2025 Toshiba Digital Solutions Corporation

07

まとめ

© 2025 Toshiba Digital Solutions Corporation

まとめ

SQLインタフェースを備えたIoT向けデータベースGridDBを使って、

- DB-API(2.0)仕様とJPype DBAPI2の利用方法について説明しました。
- また、Apache Arrowによる高速化についてもご紹介しました。

Python言語でのデータ蓄積、データ分析等にご活用ください。

GridDBのオープンソース版(GridDB CE)がありますので、是非とも使ってみてください。
https://github.com/griddb/

※本資料に掲載の製品名、サービス名には、各社の登録商標または商標が含まれています。

© 2025 Toshiba Digital Solutions Corporation

TOSHIBA

各エディションの違い

- インタフェースはほぼ同じ
- クラスタ構成の有無の違い

項目	機能	Community Edition	Enterprise Edition	Cloud
サポート			✓	✓
プロフェッショナルサービス			✓	✓
データ管理	時系列コンテナ	✓	✓	✓
	コレクションコンテナ	✓	✓	✓
	索引	✓	✓	✓
	アフィニティ	✓	✓	✓
	テーブルパーティショニング	✓	✓	✓
クエリ言語	TQL	✓	✓	✓
	SQL	✓	✓	✓
NoSQLインタフェース	Java	✓	✓	✓
	C言語	✓	✓	✓
NewSQL(SQL) インタフェース	JDBC	✓	✓	✓
	ODBC		✓	✓
WebAPI		✓	✓	✓
時系列データ	時系列分析関数	✓	✓	✓
	期限付き解放機能	✓	✓	✓
クラスタリング	機能クラスタ構成		✓	✓
	分散データ管理		✓	✓
	レプリケーション		✓	✓
運用管理	ローリングアップグレード		✓	
	オンラインバックアップ		✓	✓
	エクスポート / インポート	✓	✓	✓
	運用管理GUI		✓	✓
	CLIツール	✓	✓	✓
セキュリティ	信暗号化 (TLS/SSL)		✓	✓
	認証機能 (LDAP)		✓	
オンプレミス環境	オンプレミス環境	✓	✓	
クラウドサービス	クラウドサービス			✓

© 2025 Toshiba Digital Solutions Corporation

ご参考：GridDBに関する情報

- **GridDB GitHubサイト**
  - https://github.com/griddb/griddb/

- **GridDB デベロッパーズサイト**
  - https://griddb.net/

- **X（旧Twitter）：GridDB (日本)**
  - https://x.com/griddb_jp

- **X（旧Twitter）：GridDB Community**
  - https://x.com/GridDBCommunity
- **Facebook: GridDB Community**
  - https://www.facebook.com/griddbcommunity/
- **Wiki**
  - https://ja.wikipedia.org/wiki/GridDB
- **GridDB お問い合わせ**
  - OSS版のプログラミング関連：Stackoverflow(https://ja.stackoverflow.com/search?q=griddb)もしくはGitHubサイトの各リポジトリのIssueをご利用ください
    プログラミング関連以外：contact@griddb.netもしくはcontact@griddb.orgをご利用ください

© 2025 Toshiba Digital Solutions Corporation