日本発のオープンソース・データベースGridDB

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

2022.10.28

Contents

01

GridDBの概要

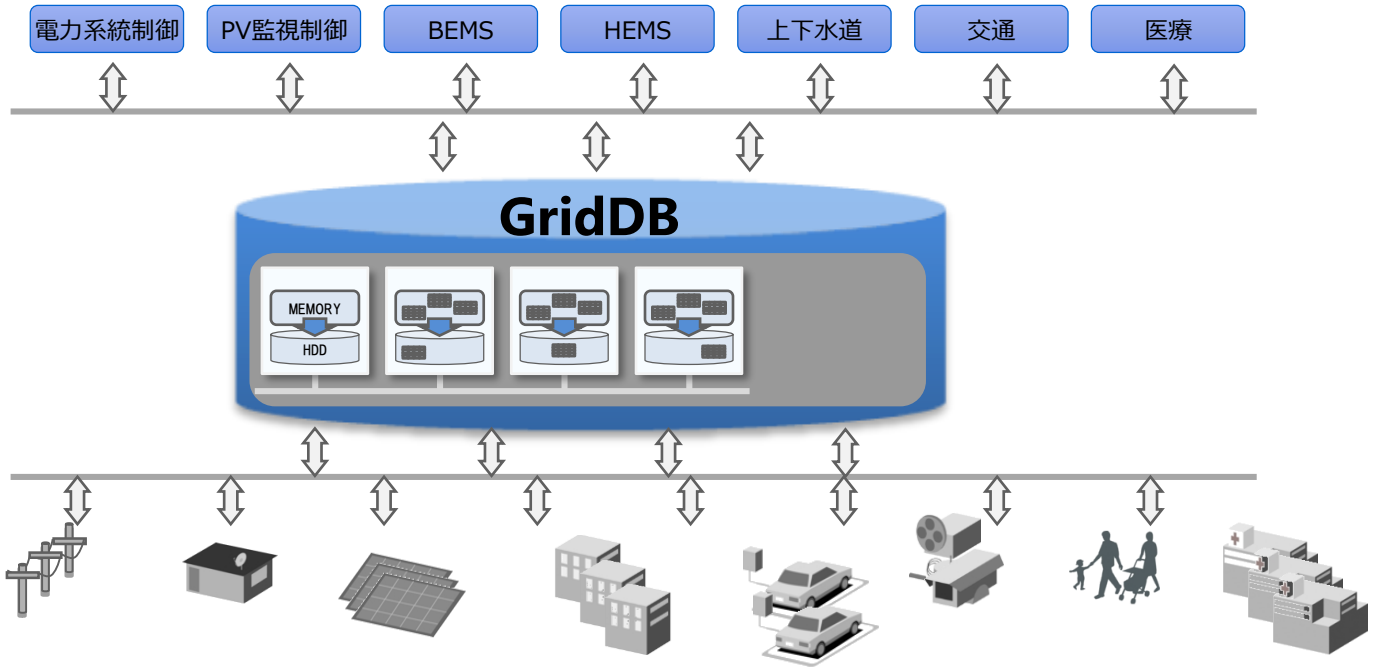
GridDBの概要

①GridDBとは？
②GridDBはオープンソース？
③オープンソース化の目的
④特徴
⑤目指す姿
⑥ユースケース

①GridDBとは？

- 日本発のビッグデータ／IoT向けデータベース

※IoT：モノのインターネット（Internet Of Things）。大量のモノ（センサなど）から得られるデータがインターネットにつながること。



②GridDBはオープンソース？

GridDB
Community 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する次世代の
オープンソースデータベース

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

社会インフラ、製造業を中心に、高い信頼性・可用性が求められるシステムに適用されている

③GridDB　オープンソース化の目的

- ビッグデータ技術の普及促進
  - 多くの人に知ってもらいたい、使ってみてもらいたい。
  - いろんなニーズをつかみたい。
- 他のオープンソースソフトウェア、システムとの連携強化

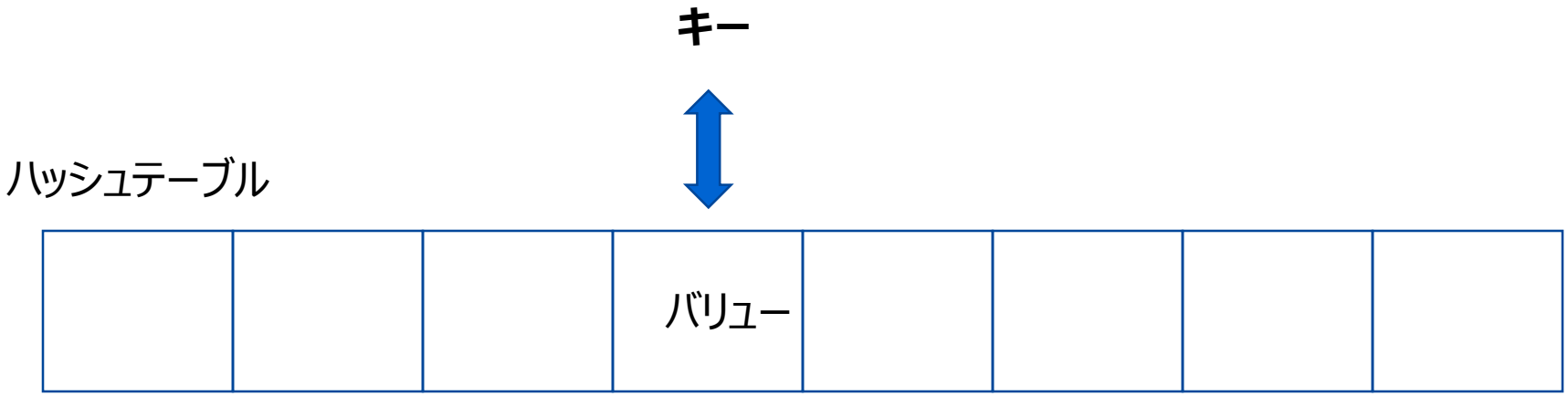
- V2.8 (2016年)
  NoSQL機能をGitHub上にソース公開
  https://github.com/griddb/griddb_nosql
- V4.5 (2020年)
  SQL機能もソース公開
  https://github.com/griddb/griddb
- 最新版 V5.1 (2022年10月25日)

④GridDB CEの特徴

時系列データ指向	• データモデルはキー・コンテナ。コンテナ内でのデーター貫性を保証 • 巨大テーブルに対するインターバル（ハッシュ）パーティショニング • パーティショニング期限解放、分析関数(SQL)
開発の俊敏性と使いやすさ	• NoSQL（キーバリュー型）インタフェースだけではなく、SQLインタフェースを提供（デュアルインタフェース） • （SQLインタフェース）ジョインなど複数テーブルに対するSQL
高い処理能力	• メモリを主、ストレージを従としたハイブリッド型インメモリDB • （SQLインタフェース）SQLにおける分散並列処理 • （NoSQLインタフェース）バッチ処理 MultiPut/MultiGet/MultiQuery
拡張性	• ペタバイト級の大規模データへの対応 • コアスケールへの対応

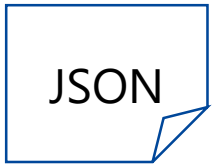
※　チェックポイント、Redoログによる耐障害性への対応

NoSQL DB (Key Value Store(KVS))とキー・コンテナモデル

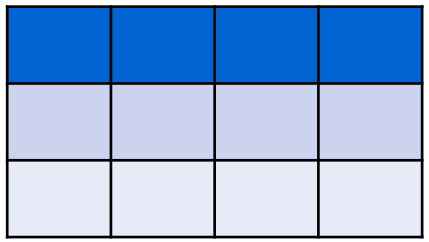


123

単純値：（例）Redis

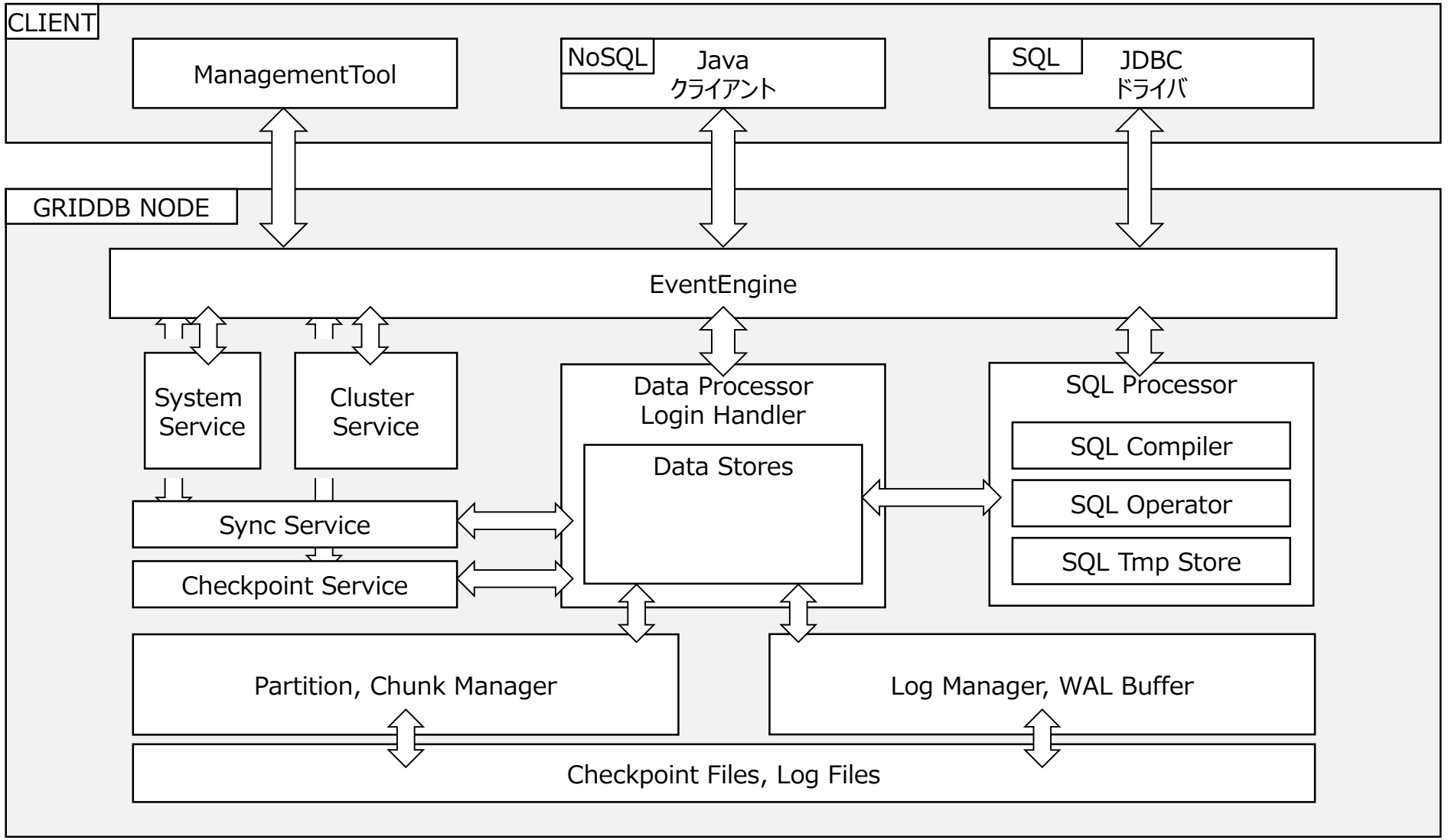


ドキュメント：（例）MongoDB

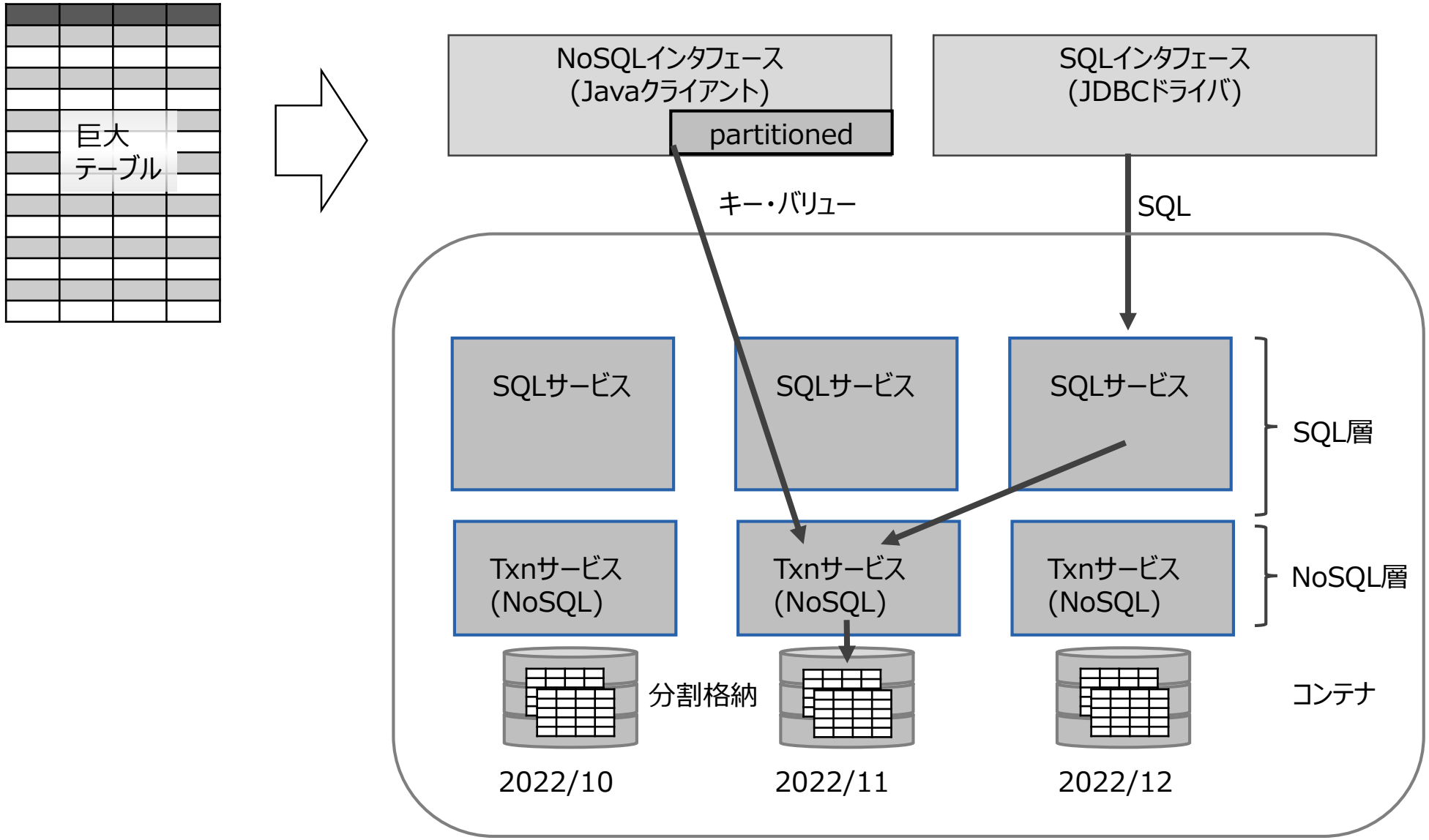


コンテナ（≒テーブル）：GridDB
※索引、検索言語TQL、トランザクションをサポート

内部モジュール構成



デュアルインタフェースとテーブルパーティショニング



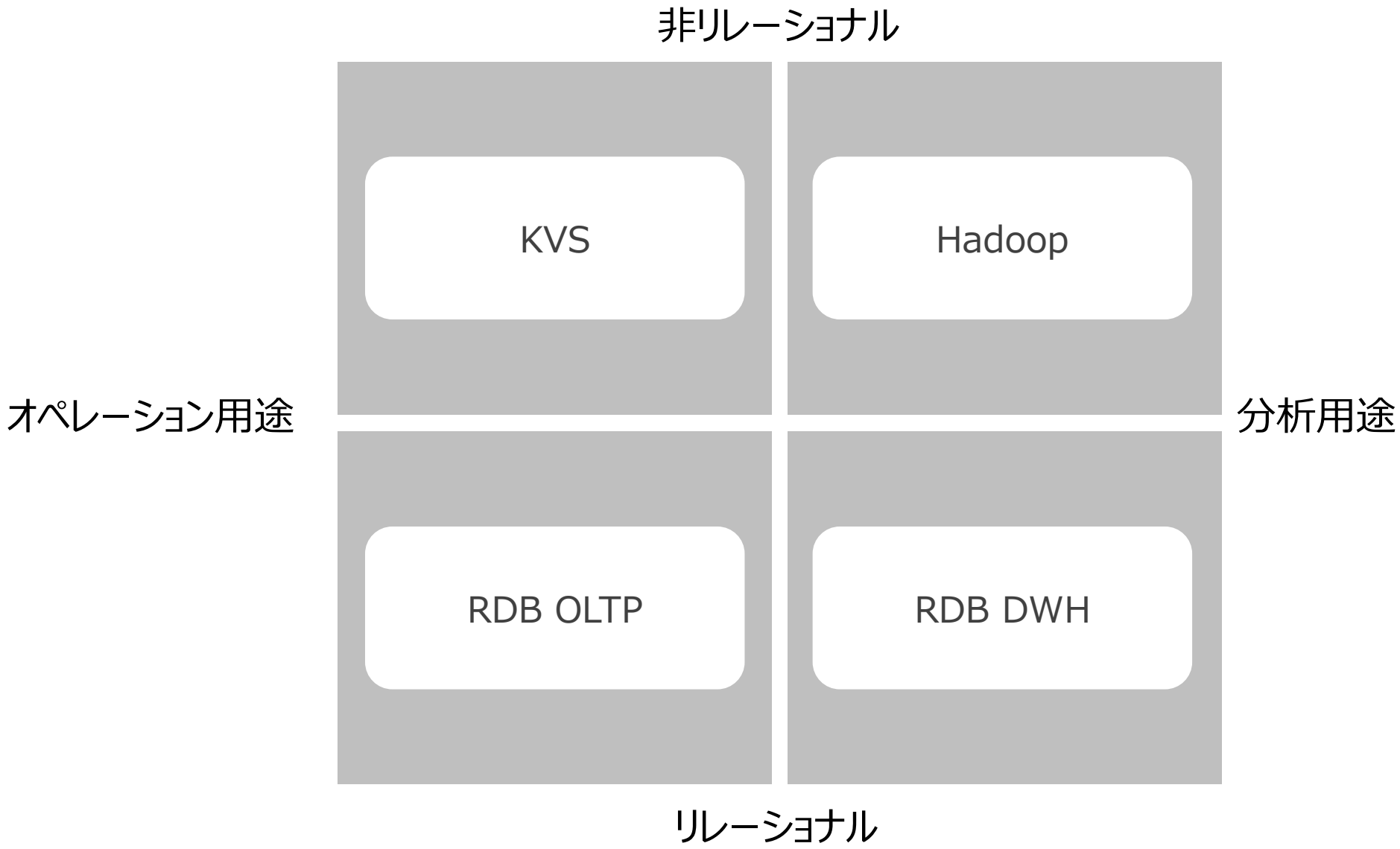
テーブルパーティショニング

- データ登録数が多い巨大なテーブルのデータを分散配置することで、プロセッサの並列実行を可能とし、巨大テーブルのアクセスを高速化するための機能
  - ハッシュパーティショニング
    - ✓ 選択基準：散らすべきキーにランダム性が高く、キーの間に処理上の関連性が無い場合
  - インターバルパーティショニング
    - ✓ 選択基準：散らすべきキーの数値的な範囲で散らしたい場合
  - インターバルハッシュパーティショニング
    - ✓ 選択基準：インターバルパーティショニングでは力不足の場合

```
-- ハッシュ
CREATE TABLE a3 (code INT, ts TIMESTAMP, dest STRING NOT NULL)
    PARTITION BY HASH(dest) PARTITIONS 10
-- インターバル
CREATE TABLE a1 (code INT, ts TIMESTAMP NOT NULL, dest STRING)
    PARTITION BY RANGE(ts) EVERY(1,DAY)
-- インターバルハッシュ
CREATE TABLE a4 (code INT NOT NULL, ts TIMESTAMP, dest STRING)
    PARTITION BY RANGE(ts) EVERY(1,DAY)
    SUBPARTITION BY HASH(dest) SUBPARTITIONS 2
```

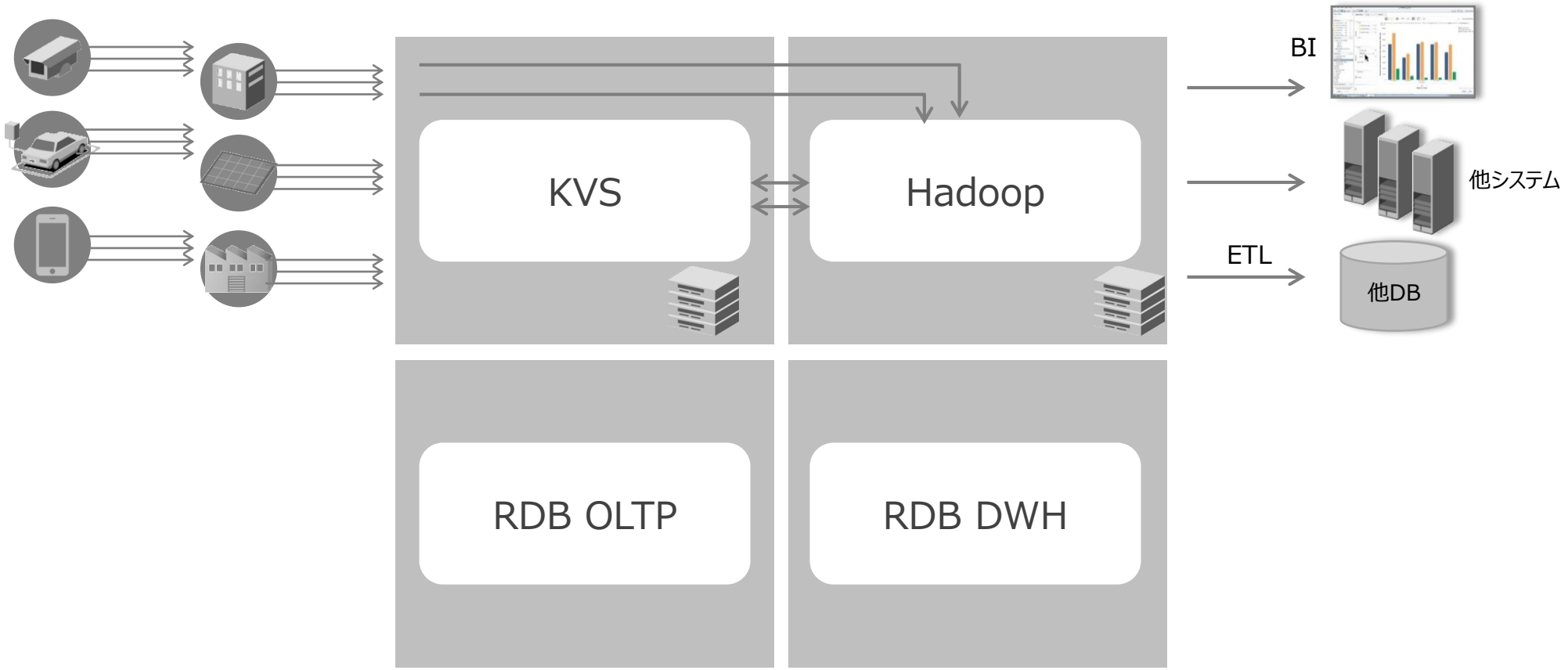
⑤GridDBの目指す姿

DB分類

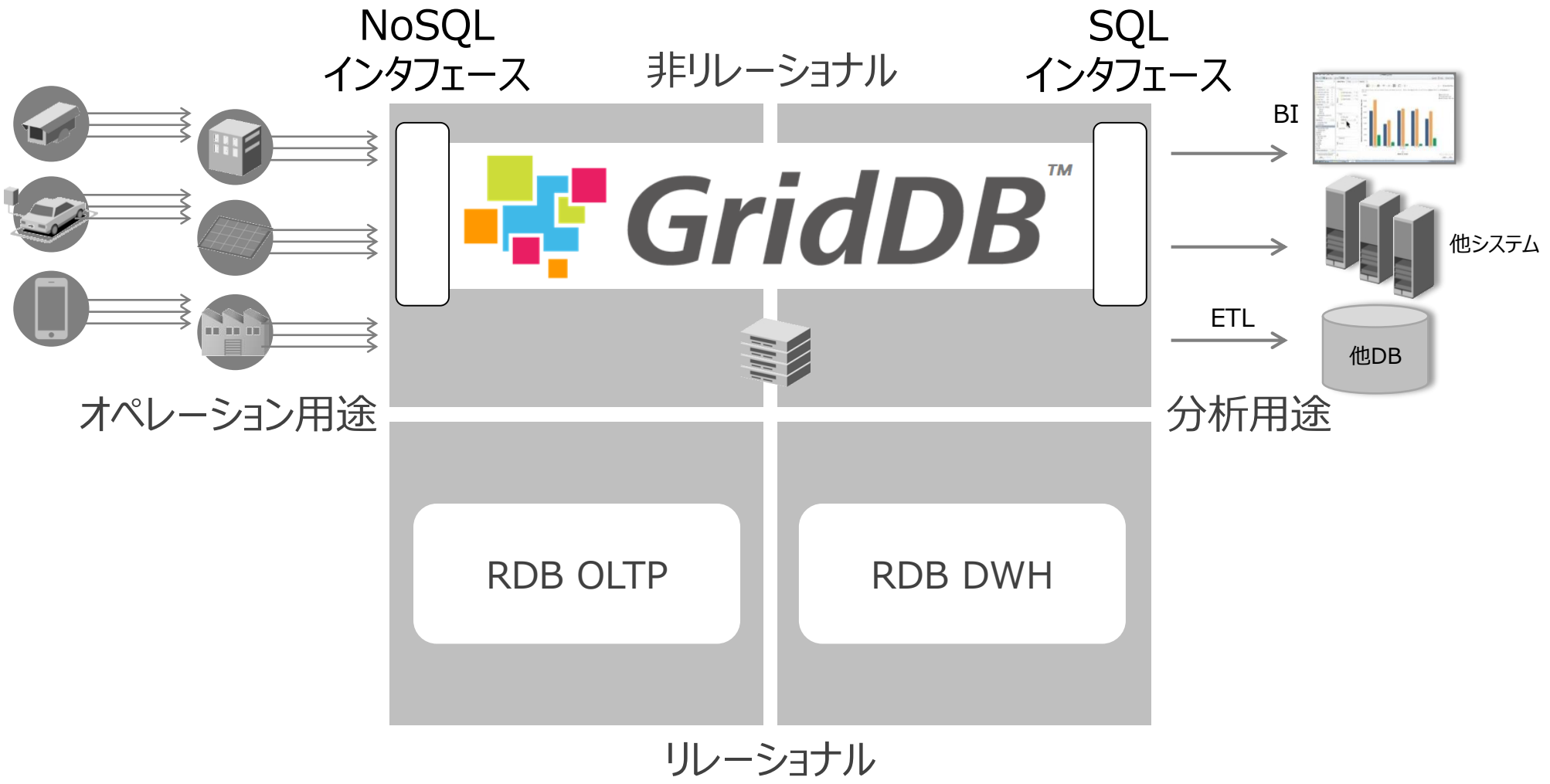


IoTシステム

（通常）収集から分析まで複数DBのサービスが必要になる



目指すもの

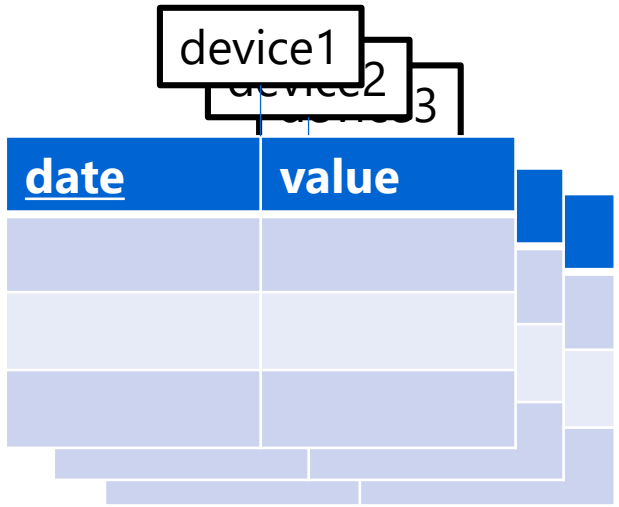


⑥ユースケース

(A) NoSQLインタフェースによるユースケース

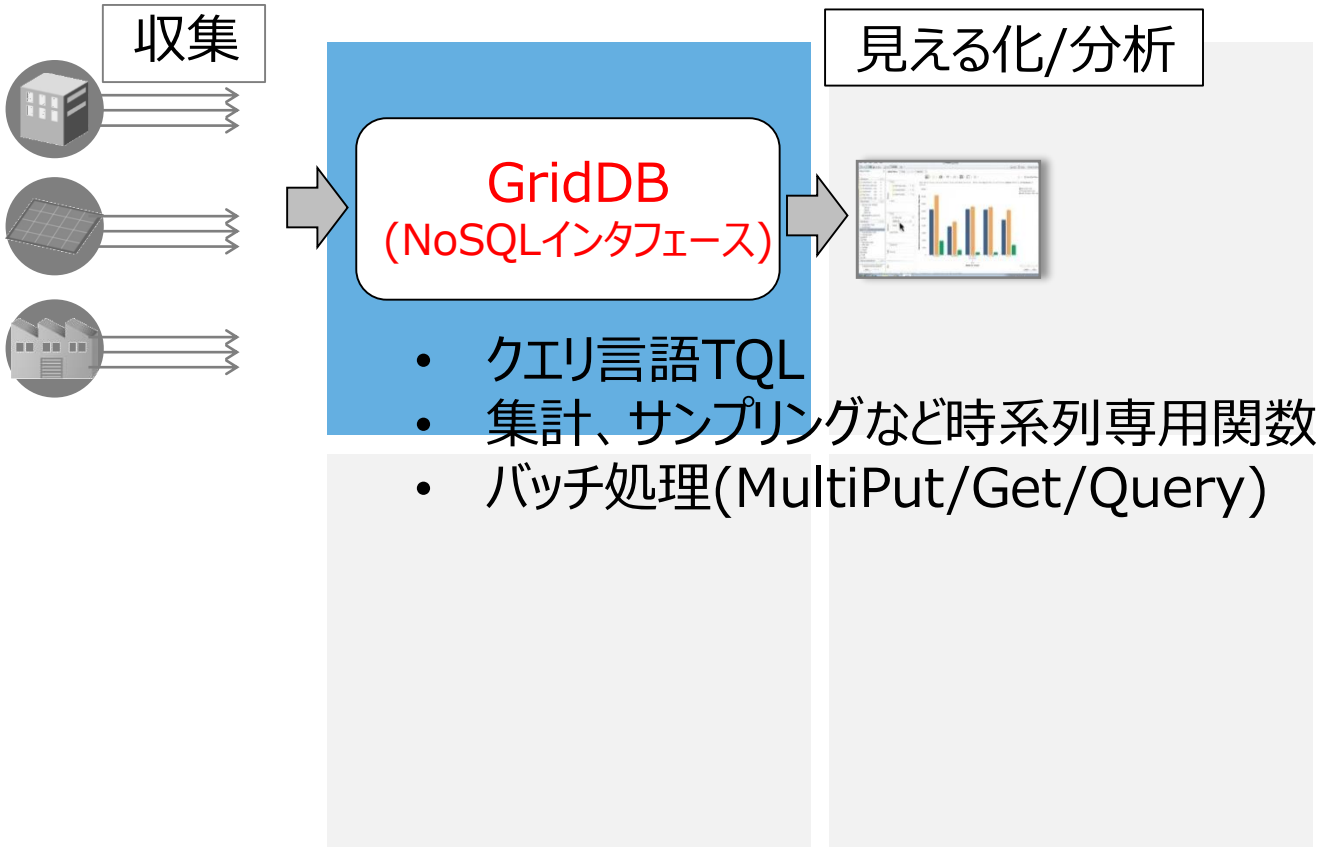
時系列データのスキーマ例（１）

装置ごとに＜日時、センサ値＞のコンテナ



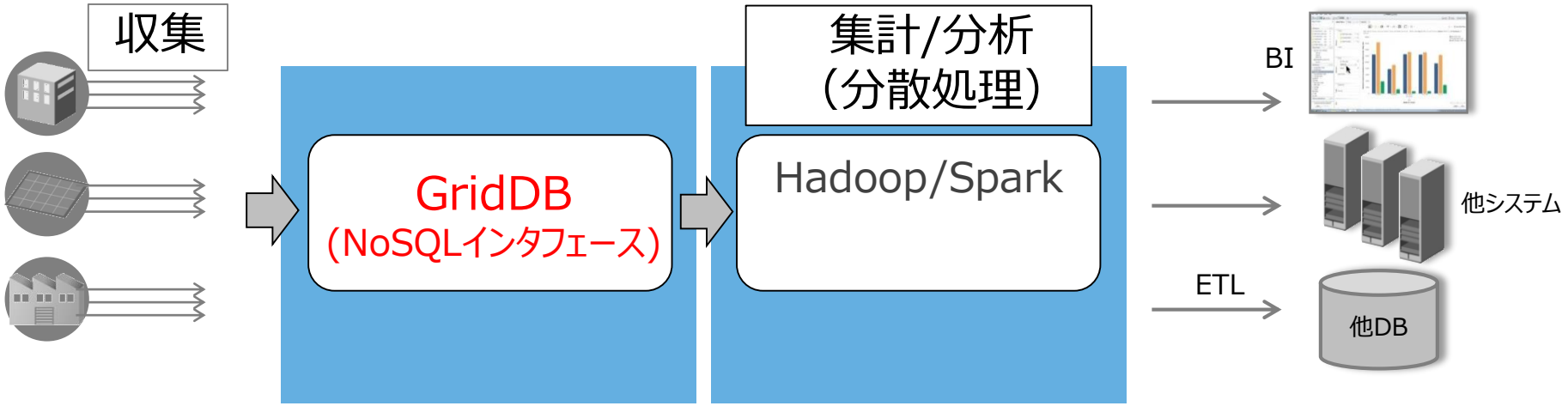

```
CREATE TABLE device1 (
 date TIMESTAMP, -- 日時
 value DOUBLE, -- センサ値);
```

A１．時系列データの管理



- 高速なシステムを実現

A２．Hadoop(Spark)による分散処理



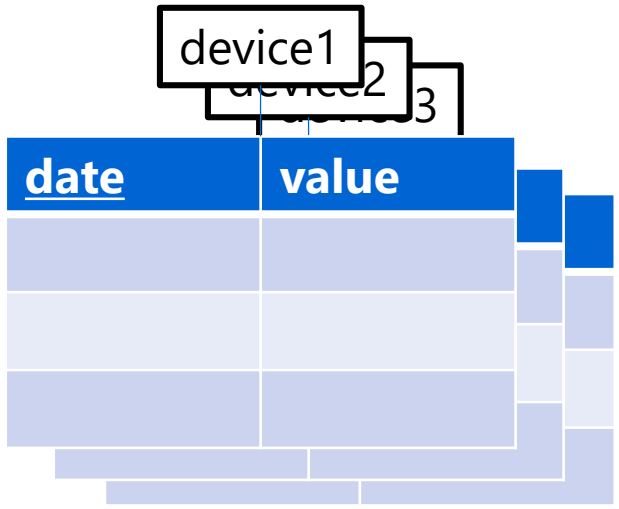
- Hadoop(Spark)を使って集計・分析を大規模分散処理
  ⇒ GridDB(NoSQLインタフェース)の特長を最大限に活かす

⑥ユースケース

(B) NoSQL/SQLのデュアルインタフェースによるユースケース

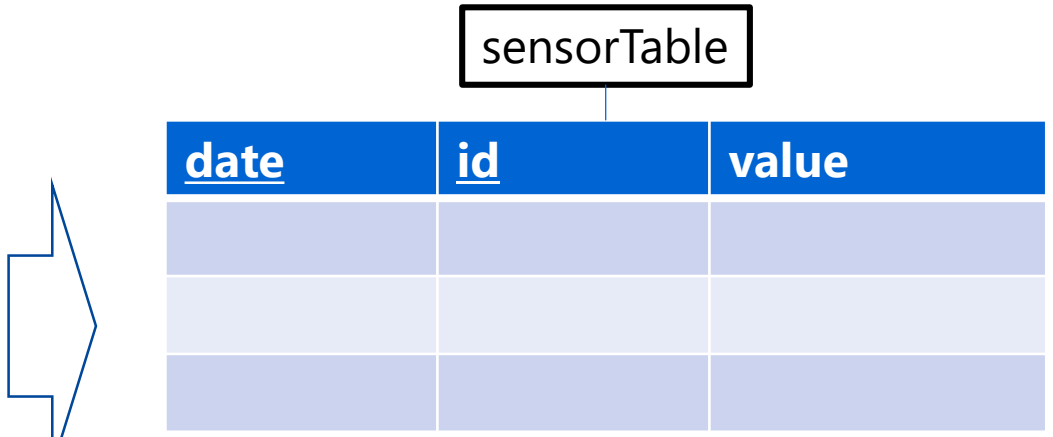
時系列データのスキーマ例（2）

装置ごとに＜日時、センサ値＞のコンテナ



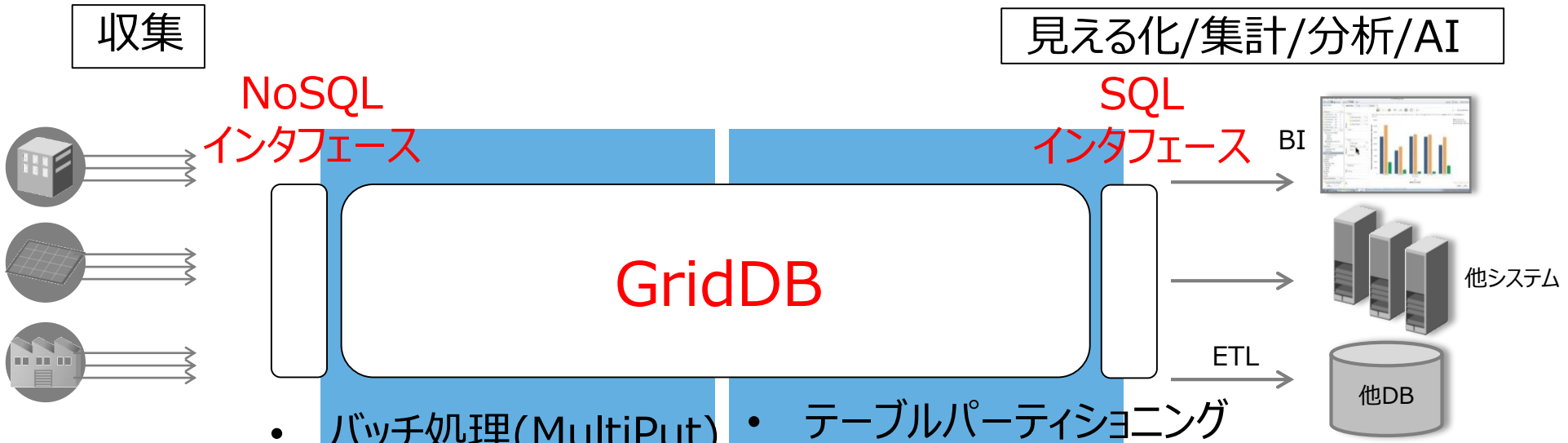

```
CREATE TABLE device1 (
 date TIMESTAMP, -- 日時
 value DOUBLE, -- センサ値);
```

＜日時、装置ID、センサ値＞のテーブル＋
インターバル（ハッシュ）パーティショニング




```
CREATE TABLE sensorTable (
  date TIMESTAMP, -- 日時
  id INTEGER, -- 装置ID
  value DOUBLE, -- センサ値
  PRIMARY KEY(date, id)
) PARTITION BY RANGE (date) EVERY (30, DAY);
SUBPARTITION BY HASH(id) SUBPARTITIONS 6;
-- 分割幅30日、サブパーティション数6の
     インターバルハッシュパーティショニング
```

B1. NoSQL/SQLデュアルインタフェースによるシステム化



- NoSQL＋SQLによる高速処理
- SQLインタフェースによる他システム連携強化

02

GridDBの利用方法

GridDBのインストール＆起動の手順 (Ubuntuの例)

【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/v5.0.0/griddb_5.0.0_am64.deb
$ sudo dpkg -i griddb_5.0.0_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/v5.0.0/griddb_cli_5.0.0_am64.deb
$ sudo dpkg -i griddb-cli_5.0.0_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
>
```

※GridDBサービスの停止

```
$ systemctl stop gridstore
```

GridDBのインストール＆起動の手順 (Ubuntuの例)

【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/v5.0.0/griddb_5.0.0_am64.deb
$ sudo dpkg -i griddb_5.0.0_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/v5.0.0/griddb_cli_5.0.0_am64.deb
$ sudo dpkg -i griddb-cli_5.0.0_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
>
```

設定なし、５つのステップだけで
CLIによるSQLなどの操作が開始できる。

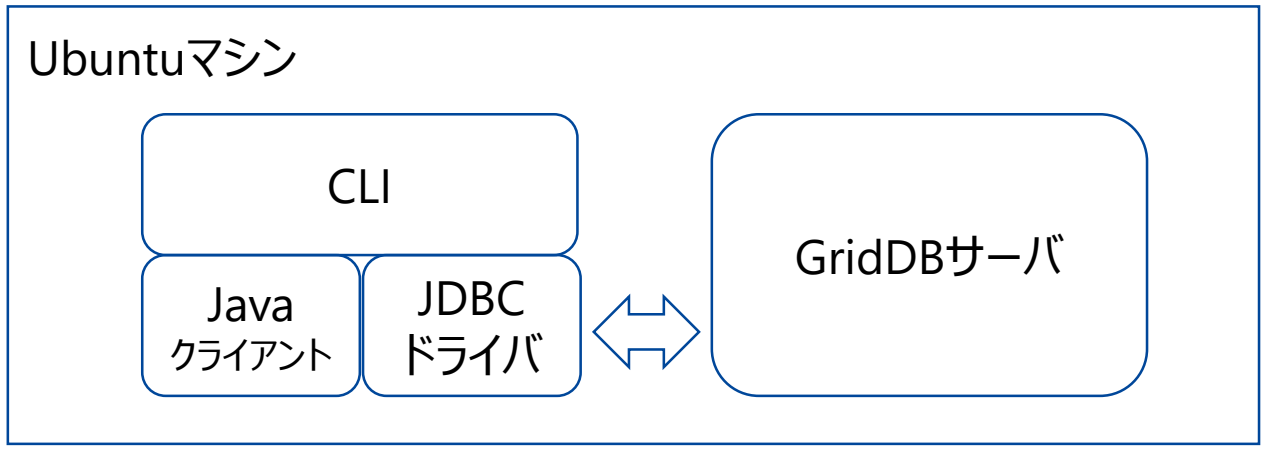
※GridDBサービスの停止

```
$ systemctl stop gridstore
```

＜動作環境の前提条件＞

- Windows10マシン上にVirtualBox
- ゲストOSはUbuntu 22.04。Java 11(デフォルト)インストール済

- 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、Javaクライアント：https://github.com/griddb/griddb
※GridDB JDBCドライバ： https://github.com/griddb/jdbc
※GridDB CLI：https://github.com/griddb/cli

実行例１（SQL基本）

```
# 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
```

※SQL文の先頭が下記文字列のいずれかである場合、コマンド名sqlを省略することができます。
select update insert replace delete create drop alter grant revoke pragma explain

実行例２（テーブルパーティショニング）：テーブル作成

装置

id	type	floor	room_no

```
CREATE TABLE equipTable (
  id INTEGER PRIMARY KEY, -- 装置ID
  type STRING,   -- 装置タイプ
  floor INTEGER, -- 設置階
  room_no INTEGER -- 設置ルームNo
);
```

センサデータ

date	id	value
インターバルハッシュパーティション： 分割幅30日、サブパーティション数6 パーティション解放：60日		

```
CREATE TABLE sensorTable (
  date TIMESTAMP, -- 日時
  id INTEGER, -- 装置ID
  value DOUBLE, -- センサ値
  PRIMARY KEY(date, id)
) WITH (
  expiration_type='PARTITION',
  expiration_time=60,
  expiration_time_unit='DAY'
) PARTITION BY RANGE (date) EVERY (30, DAY);
SUBPARTITION BY HASH(id) SUBPARTITIONS 6;
```

実行例２（テーブルパーティショニング）：データの登録

装置

id	type	floor	room_no
1	CAMERA	1	1
2	THERMO	1	1
｡｡｡			

センサデータ

date	id	value
2021-11-01T10:30:00Z	2	18.5
2021-11-01T10:30:00Z	3	20.0
｡｡｡		

```
INSERT INTO equipTable VALUES(1, 'CAMERA', 1, 1);
INSERT INTO equipTable VALUES(2, 'THERMO', 1, 1);
INSERT INTO equipTable VALUES(3, 'THERMO', 4, 3);
INSERT INTO equipTable VALUES(4, 'THERMO', 6, 2);
INSERT INTO equipTable VALUES(5, 'WATT', 1, 1);
INSERT INTO equipTable VALUES(6, 'WATT', 6, 1);
```

```
INSERT INTO sensorTable
  VALUES(TIMESTAMP('2021-11-01T10:30:00Z'), 2, 18.5);
INSERT INTO sensorTable
  VALUES(TIMESTAMP('2021-11-01T10:30:00Z'), 3, 20.0);
｡｡｡
```

ご参考：

- SQL（テーブルパーティショニング）の例
  - ✓ https://github.com/knonomura/griddb-docker/blob/master/SQLSamples.md
  - ✓ https://github.com/knonomura/griddb-docker/blob/master/SQLSamples2.md

- NoSQLインタフェースでバッチ処理等を使いたい場合
  - ✓ https://github.com/griddb/griddb/tree/master/sample/guide/ja

  のSampleMultiPut.javaなどを参照願います。

- リモートマシンから操作したい場合
  - ✓ https://github.com/griddb/griddb/blob/master/README_ja.md
  - ✓ https://github.com/griddb/griddb/blob/master/docs/TroubleShootingTips_ja.md

  を参照願います。

- DockerでGridDBを使いたい場合
  - ✓ https://github.com/griddb/griddb-dockerのDockerfile
  - ✓ https://hub.docker.com/u/griddbのDockerイメージ

  を参照願います。

03

OSS活動

主なOSS活動

① GridDB本体の機能強化

② 主要OSSとの連携強化

③ APIの拡充

④ GitHub以外のサイトからの情報発信

- パッケージ
- デベロッパーズサイト（WP、ブログなど）・・・フィックスターズ社
- SNS・・・フィックスターズ社

⑤ 主要OSSリポジトリへのコントリビュート

⑥ プラットフォームの拡充

⑦ その他

- OSCなどカンファレンス参加
- ハンズオン無料セミナー・・・（株）アイ・ティ・イノベーション

OSS活動の全体イメージ

性能測定 | 収集 | 分散処理 | 可視化 | Webアプリ | 分析 | AI/機械学習 …

④GitHub以外のサイトからの情報発信
PyPI/npm/Maven/Packagist/…

⑥プラットフォームの拡充
CentOS＋
Ubuntu、openSUSE
Windows、MacOS
Docker

②主要OSSとの連携強化

Spark コネクタ
Fluentd/Grafana/Redash プラグイン
YCSB コネクタ
Kafka コネクタ
Hadoop MapReduce コネクタ

Rust クライアント

WebAPI
Python/Node.JS/Go/PHP/Ruby/Perlクライアント

③APIの拡充

Javaクライアント
JDBCドライバ
Cクライアント

①GridDB本体の機能強化
GridDB V5.1 CE (Community Edition)

https://github.com/griddb/
GitHub

⑤主要OSSリポジトリへのコントリビュート

最近の主な活動（2022年度）

2022年

- **4月　　V5.0CEのソース公開**
- **9月　　Apache Kafkaコネクタのソース公開**
- **10月　　Rust言語のクライアントライブラリのソース公開**
  V5.1CEのソース公開

最近の改良点

＜V5.0CE＞

- 大規模データに対する性能強化
  - ✓ コアスケール対応
  - ✓ スキャン、データ削除高速化
  - ✓ チェックポイント実行時のディスクI/O負荷低減
- 機能強化
  - ✓ SQLによるカラム名変更
  - ✓ GridDBサービス機能
  - ✓ Exp/Impツール

＜V5.1CE＞

- 機能強化
  - ✓ クライアント通信経路設定

デベロッパーズサイト

https://griddb.net/

griddb net | 検索

- **アプリケーション開発者向けのサイト**
- **様々なコンテンツを公開**
  - ホワイトペーパ
  - ブログ

  など

ツイッター

https://twitter.com/griddb_jp

twitter griddb	検索

- GridDBに関するリリース、イベント、などをお知らせします。
  (日本国内向け)

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/
- **Twitter: GridDB (日本)**
  - https://twitter.com/griddb_jp
- **Twitter: GridDB Community**
  - https://twitter.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

griddb github	検索

griddb net	検索

twitter griddb	検索

04

まとめ

まとめ

- GridDBはビッグデータ・IoT向けのデータベースです。
- GridDBの概要と使い方、オープンソース活動についてご紹介しました。
  - 今後も様々な拡張、拡充を進めて参ります。

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

TOSHIBA

付録

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			✓

CLIの主なコマンド一覧 （１）

変数定義

コマンド	引数	説明
setnode	[テキストファイル名]	ノード変数を定義します。
setcluster	クラスタ変数名 クラスタ名 マルチキャストアドレス　ポート番号 [ノード変数...]	クラスタ変数を定義します。
setclustersql	クラスタ変数名 クラスタ名 SQLアドレス　SQLポート番号	クラスタ構成にSQLの接続先を定義します。
setuser	ユーザ名 パスワード [OSユーザgsadmのパスワード]	クラスタにアクセスするユーザおよびパスワードを定義します。
show	[変数名]	変数の定義内容を表示します。
save	[スクリプトファイル名]	変数定義をスクリプトファイルに保存します。
load	[スクリプトファイル名]	スクリプトファイルを読み込み、実行します。

接続・切断

コマンド	引数	説明
connect	クラスタ変数 [データベース名]	GridDBクラスタに接続します。
disconnect		GridDBクラスタから切断します。

CLIの主なコマンド一覧 （２）

コンテナ操作

コマンド	引数	説明
createcollection	コンテナ名 カラム名　タイプ [カラム名　タイプ ...]	コンテナ(コレクション)を作成します。
dropcontainer	コンテナ名	コンテナを削除します。
putrow	コンテナ名 値 [値...]	コンテナにロウを登録します。
removerow	コンテナ名 ロウキー値 [ロウキー値...]	コンテナのロウを削除します。
createindex	コンテナ名　カラム名　索引タイプ...	指定カラムに索引を作成します。
tql	コンテナ名　クエリ　;	TQL検索を実行し、検索結果を保持します。
sql	SQL文　;	SQL文を実行し、検索結果を保持します。
get	[取得件数]	検索結果を取得し、標準出力に表示します。
tqlexplain	コンテナ名　クエリ　;	指定TQL文の実行計画を表示します。

※SQL文の先頭が下記文字列のいずれかである場合、コマンド名sqlを省略することができます。
select update insert replace delete create drop alter grant revoke pragma explain

CLIの主なコマンド一覧 （３）

ステイタス・一覧表示

コマンド	引数	説明
showcontainer (showtable)	[コンテナ名(テーブル名)]	コンテナ(テーブル)情報を表示します。
searchcontainer	[コンテナ名]	コンテナ名からコンテナを検索します。
showsql	[クエリID]	実行中のSQL処理を表示します。
showevent		実行中のイベント一覧を表示します。
showconnection		コネクションの一覧を表示します。

実行計画

コマンド	引数	説明
getplantxt	[テキストファイル名]	実行計画をテキスト形式で表示します。
getplanjson	[JSONファイル名]	実行計画をJSON形式で表示します。

Javaクライアントによるバッチ処理

```
List<String> containerNameList = new ArrayList<String>();
…
final Map<String, List<Row>> rowListMap = new HashMap<String, List<Row>>();
Random rnd = new Random();
for (String containerName : containerNameList) {
        final List<Row> rowList = new ArrayList<Row>();
        ContainerInfo containerInfo;
        containerInfo = store.getContainerInfo(containerName);
        SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.SSS");
        Date tm1=sf.parse("2015-04-27 19:00:00.000");
        for (int j = 0; j < rowCount; j++) {
                Row row = store.createRow(containerInfo);
                row.setTimestamp(0,TimestampUtils.add(tm1, j, TimeUnit.MINUTE));
                row.setBool(1, false);
                row.setDouble(2, rnd.nextInt(10000));
                rowList.add(row);
        }
        rowListMap.put(containerName, rowList);

}
store.multiPut(rowListMap);
```

※https://github.com/griddb/griddb/tree/master/sample/guide/ja
のSampleMultiPut.javaの抜粋

リモートマシンからのアクセスを可能にするには

1. gsadmユーザでログイン
$ su – gsadm
2. コンフィグ変更（デフォルトはローカル接続限定の設定になっているため）
$ cp /usr/griddb-X.X.X/conf_multicast/* conf/.
3. GridDB管理者(admin)のパスワード設定
$ gs_passwd admin
> admin
4. GridDBサーバの起動、（1ノードでの）クラスタ構成
$ gs_startnode
$ gs_joincluster –u admin/admin –c myCluster

・Stat確認
$ gs_stat –u admin/admin