



# Amazon Timestream

## 新機能紹介と実演

Yosuke Nishihara

アマゾン ウェブ サービス ジャパン合同会社

2023/09

# Agenda

1. バッチロード
2. AWS Backup との統合
3. アンロード
4. ユーザ定義のパーティションキー

# 1. バッチロード

S3 に保存した CSV ファイルを直接 Amazon Timestream に取り込めるようになり、他データベースからのデータ移行がプログラミングレスで容易に実装できる。ジョブの並列実行により高速なローディングを実現。

## 1. データファイルの準備

```
id1,id2,timestamp,m_name,val1,val2
"009","K","1679273577275","measure","27.8","-21.5"
"006","A","1679273577276","measure","13.4","21.5"
"005","H","1679273577278","measure","-11.9","-13.7"
"001","B","1679273577279","measure","8.3","6.5"
"002","M","1679273577288","measure","-4.1","-9.1"
...
```

S3 に CSV ファイルを配置。メジャー値とディメンジョン値が最低 1 つは必要。また、タイムスタンプは UNIX 時間で記述し、ファイルサイズは 5GB まで。

## 2. データマッピングを記述

ビジュアルビルダー (7) 情報 すべてのマッピングをリセット 列マッピングを削除 列マ

🔍 列名で検索

<input type="checkbox"/>	ソース列名	ターゲットテーブルの列名	Timestream 属性タイプ	データ型
<input type="checkbox"/>	city	city	DIMENSION	VARCHAR
<input type="checkbox"/>	device	device	DIMENSION	VARCHAR
<input type="checkbox"/>	initial	initial	DIMENSION	VARCHAR
<input type="checkbox"/>	timestamp	time	TIMESTAMP	TIMESTAMP
<input type="checkbox"/>	measure_name	measure_name	MEASURE_NAME	-
<input type="checkbox"/>	val1	val1	MULTI	VARCHAR
<input type="checkbox"/>	val2	val2	MULTI	VARCHAR

ジョブの作成時にマッピングを定義。ロード先のテーブルとともに、各カラムと属性タイプ、データ型を指定。

## 3. ジョブ実行

Timestream > バッチロードタスク > [タスク名]

バッチロードタスク: [タスク名]

概要

タスク ID	作成時刻
[タスク ID]	March 20, 2023, 13:36:13 (UTC+09:00)
ステータス	最終更新
🟢完了済み	March 20, 2023, 13:36:20 (UTC+09:00)

進捗報告

最終更新: March 20, 2023, 13:36:20 (UTC+09:00)

レコードが処理されました	レコードの取り込みの失敗
9	-
取り込まれたレコード	取り込まれたバイト数
9	1024
ファイル解析の失敗	ファイルの失敗
-	-

ジョブ実行後、レコードの処理数やエラー発生数、エラー理由等が確認出来る。

# デモの内容

- Amazon S3 にテスト用データを配置
- Visual Editor を使ったデータマッピングを実施
- ロードの実行
- データの確認

# 使用するデータ

ディメンジョン

メジャー

time	deviceId	location	measure_name	temperature	humidity
2023-09-05 19:00:00.000498	394	東京	metrics	32.1	80
2023-09-05 19:00:01.020958	1543	大阪	metrics	35.1	81
2023-09-05 19:00:02.031958	9873	福岡	metrics	33.7	76
2023-09-05 19:00:03.487059	6819	奈良	metrics	35.8	80
2023-09-05 19:00:03.261059	49	佐賀	metrics	31.5	67
2023-09-05 19:00:03.994651	9200	京都	metrics	36.9	81
2023-09-05 19:00:03.019263	1048	北海道	metrics	24.1	54
2023-09-05 19:00:04.138948	99	埼玉	metrics	34.3	79
2023-09-05 19:00:04.713981	5081	高知	metrics	36.4	75
2023-09-05 19:00:05.669690	7756	徳島	metrics	35.5	74
...	...	...	...	...	...

## Amazon Timestream

### ダッシュボード

#### ▼ リソース

データベース

テーブル

バックアップ **新規**

#### ▼ 管理ツール

バッチロードタスク **新規**

クエリエディタ

スケジュールされたクエリ

**新規**

モニタリング

#### ▼ 学習リソース

Timestream > ダッシュボード

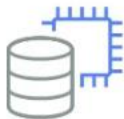
## ダッシュボード 情報

デフォルトレイアウトにリセット

+ ウィジェットを追加

### Amazon Timestream の開始方法

Get started with Amazon Timestream by following these steps.



#### 1. サンプルデータベースとテーブルを作成する

サンプルデータベースとテーブルにはサンプルデータが付属しているため、簡単に使い始めることができます。



#### 2. サンプルクエリを試してください

クエリエディターで待機しているサンプルクエリのいずれかを使用して、サンプルテーブルのデータを表示してください。



#### 3. データの取り込みを開始する

Amazon Timestream は、さまざまな AWS サービスや一般的なサードパーティーツールと統合できるため、データ移行をシームレスに行うことができます。

## 特記事項

- CSV ファイルは 100MB ~ 1GB 程度が望ましい
- 並列にジョブを実行する事で、パフォーマンス改善が期待出来る
- バッチロードタスクの実行中は、同じテーブルへのデータ書き込みはタスクのエラーにつながる可能性があります、避けた方が良い
- バッチロードタスクの実行中は、対象の S3 バケットのファイルの変更、削除はタスクエラーの原因となります

## ユースケース

- 別 DB から Timestream へデータ移行
- オンライン停止時間帯でのデータコピー
- 検証作業

## 2. AWS Backup との統合

AWS Backup との統合により、フルマネージドで Timestream のバックアップを取得・管理できるようになった

- 表単位でのバックアップの作成
  - オンデマンド、もしくはスケジューリングによる自動取得が可能
  - Incremental Backup をサポート
- データの復元
  - ソースとは独立した新規の表として復元
  - **リージョンやアカウントを跨いだ復元 (複製) も可能**
  - 全てのデータの復元、もしくは特定期間のデータを選択した復元が可能





## デモの内容

- AWS Backup で Timestream テーブルのバックアップを実行
- 取得したバックアップを使ってテーブルを復元

## AWS Backup

### ▼ マイアカウント

ダッシュボード

バックアップポータル

バックアップ保管庫ロック

バックアッププラン

**保護されたリソース**

ジョブ

リーガルホールド

設定

### ▼ 外部リソース

ゲートウェイ

ハイパーバイザー

仮想マシン

## AWS Backup > 保護されたリソース

### 保護されたリソース (3) 情報

AWS Backup によってバックアップされるリソース

オンデマンドバックアップを作成

🔍 名前、ID、タイプ、または最終バックアップでリソースを検索する

< 1 > ⚙️

リソース名	リソース ID	リソースタイプ	最後のバックアップ
database/testdb/table/test_t	<a href="#">table/test_t</a>	Timestream	2023年9月9日, 8:13 (UTC+09:00)
database/testdb/table/resultTable	<a href="#">table/resultTable</a>	Timestream	2023年9月7日, 14:00 (UTC+09:00)
database/testdb/table/test	<a href="#">table/test</a>	Timestream	2023年9月7日, 14:00 (UTC+09:00)

### 3. アンロード

SQL の実行結果を **Parquet/CSV フォーマットで S3 に出力**できる。出力データの圧縮や、パーティション化、暗号化にも対応。Amazon Athena や Amazon Redshift を利用した時系列データの分析を簡単に実装できる。

```
UNLOAD
```

```
(  
  SELECT user_id, measure_name, time, quantity, product_id, channel  
  FROM sample_clickstream.sample_shopping  
  WHERE time BETWEEN ago(2d) AND now()  
)
```

```
TO 's3://<bucket_name>/partitionbychannel/'
```

```
WITH
```

```
(  
  partitioned_by = ARRAY ['channel'],  
  format='CSV',  
  compression='GZIP'  
)
```

← S3 に出力する SQL 文を記述

← 出力先の S3 のパスを記述

各種オプションを指定

- 指定したカラムでのパーティション化
- 出力フォーマット
- 圧縮有無 等

# デモの内容

- UNLOAD の実行
- S3 に出力されたデータの確認



エディタ

最近

保存されたクエリ

サンプルクエリ

データベース



Choose a database to query.

testdb

Tables (2)

Filter tables

test\_t

location (varchar)

Query 1

Query 2

Query 3

```

1 UNLOAD(
2 select * from testdb.test_t
3 where deviceid = '1111'
4 order by time
5 limit 10
6 )
7 TO 's3://timestream20230914/'
8

```



# UNLOAD 設定オプション

オプション名	内容	デフォルト
PARTITIONED_BY	パーティションの為のカラム名を ARRAY 句と併せて指定する	-
FORMAT	出力ファイルフォーマット (CSV または PARQUET)	CSV
COMPRESSION	圧縮有無 ( GZIP または NONE )	GZIP
ENCRYPTION	暗号化指定 ( SSE_KMS または SSE_S3 )	SSE_S3
KMS_KEY	顧客管理の KMS 鍵を記載	-
FIELD_DELIMITER	区切り文字	カンマ
ESCAPED_BY	エスケープ文字	バックスラッシュ

## UNLOAD 利用時の注意点

- タイムアウト (60分) を考慮すると、出力データのサイズは 60 GB 以内に抑えた方が良い
- UNLOAD が何らかのエラーで終了した場合には、不完全なデータが S3 に出力される可能性有
- partition\_by 句で指定するカラムは、参照クエリの最後に指定し、データは ASCII のみ許容。  
また、101 個以上のパーティションは作成出来ない。

# 4. ユーザ定義のパーティションキー

ディメンジョンをテーブルのパーティションのキーとして個別に設定できるようになり、設計時の選択肢が広がった

## スキーマの設定 - 新規 情報

スキーマは、テーブルに必要なデータモデルを指定します。パーティションキーを使用してデータを保存および配布する方法を指定します。カスタムパーティションキーを使用すると、クエリをより迅速かつ効率的に実行できます。

### パーティションキー設定

カスタムパーティションキーを作成するか、または Timestream のデフォルトパーティションキーを使用するかを選択します。

#### カスタムパーティショニング

これは、クエリの主なアクセスパターンが、カーディナリティの高いディメンジョンまたはメジャーをフィルタリングする場合に役立ちます。

#### デフォルトパーティショニング

これは、クエリの主なアクセスパターンが測定ベースのフィルタリングである場合に役立ちます。  
[measure\_name] 属性の値は、データをパーティショニングするためのパーティションキーとして使用されます。

## 複合パーティションキー 情報

テーブルデータのパーティショニングに使用される属性を定義するパーティションキーのリスト。リストの順序によってパーティショニング階層が決まります。

### 1-パーティションキータイプ

ディメンジョン ▼

### パーティションキー名

deviceid

#### レコードでパーティションキーを強制

新しいレコードを取り込む際には、ディメンジョンキーを指定する必要があります。

④ テーブルを作成した後に、パーティションキーを追加したり、削除したりすることはできません。キーは、テーブルが作成された後にテーブルの詳細ページで確認できます。

## 特徴

- 以下の条件に合致する場合、クエリのパフォーマンス改善に有用
  - デフォルトの measure\_name でのパーティショニングでは目標のクエリパフォーマンスを達成出来ない
  - カーディナリティの高いディメンジョンがあり、かつ、改善が必要なクエリの where 条件に対象のディメンジョンが含まれる
- パーティションキーが必須かどうかをキー作成時に指定可能

## 注意点

- テーブル作成後にはパーティションキーを作成、変更、削除する事が出来ない為、入念な事前確認が必要

# Amazon Timestream の構成要素と制約

ディメンジョン

メジャー

time	deviceId	location	measure_name	temperature	humidity	...
2023-09-05 19:00:00	49820	東京	metrics	32.1	80	
2023-09-05 19:00:01	1543	大阪	metrics	35.1	81	
2023-09-05 19:00:02	59873	福岡	metrics	33.7	76	
2023-09-05 19:00:03	6819	奈良	metrics	35.8	80	
2023-09-05 19:00:03	49	佐賀	metrics	31.5	67	
2023-09-05 19:00:03	9200	京都	metrics	36.9	81	
2023-09-05 19:00:03	1048	北海道	metrics	24.1	54	
2023-09-05 19:00:04	99	埼玉	metrics	34.3	79	
2023-09-05 19:00:04	439081	高知	metrics	36.4	75	
2023-09-05 19:00:05	4958	徳島	metrics	35.5	74	

入力可能なメジャー値は最大 256 個

最大で 8,192 種類の値



## ■ シングルメジャーレコード

単一のメジャーを1つのテーブル行に格納するデータモデル

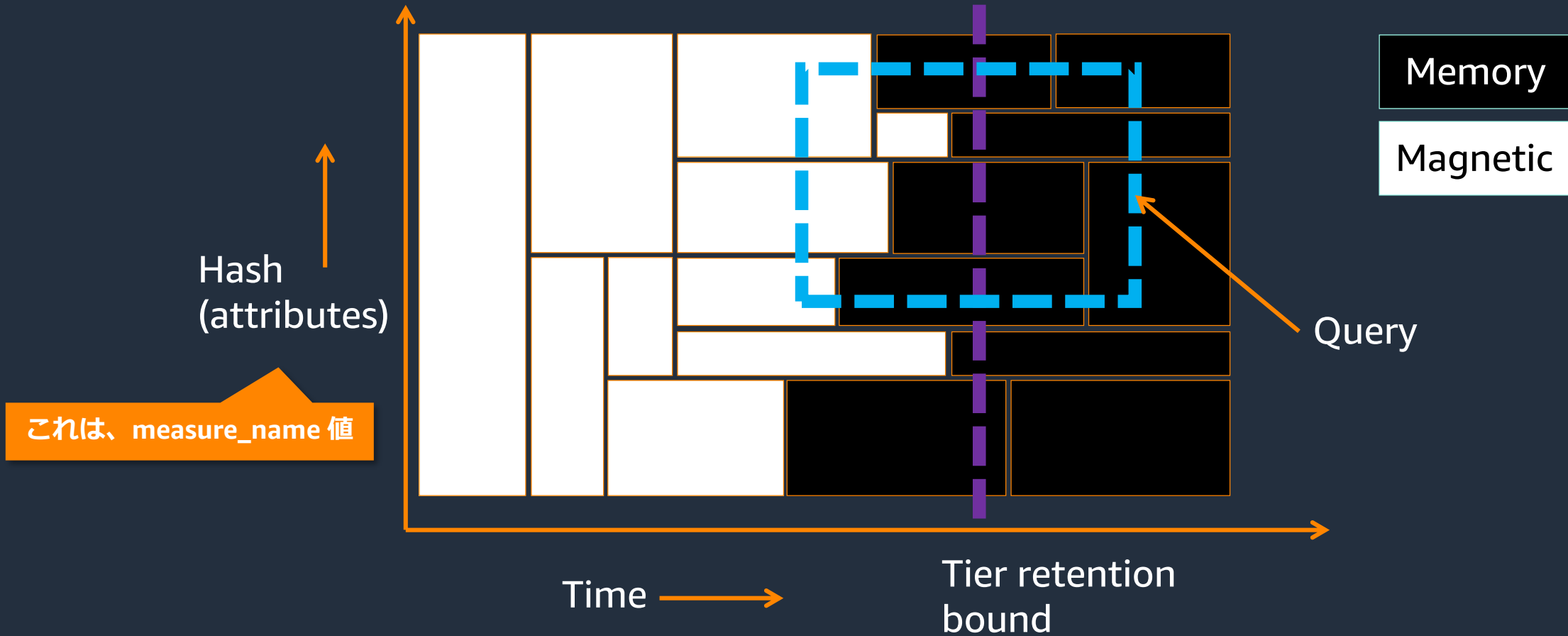
time	deviceId	location	measure_name	measure_value:double
2023-09-05 19:00:00	49820	東京	temperature	32.1
2023-09-05 19:00:00	49820	東京	humidity	80
2023-09-05 19:00:02	59873	福岡	temperature	33.7
2023-09-05 19:00:02	59873	福岡	humidity	76
2023-09-05 19:00:02	59873	福岡	temperature	33.7
2023-09-05 19:00:02	59873	福岡	humidity	76

## ■ マルチメジャーレコード (re:Invent2021 より追加されたデータモデル)

複数のメジャーを1つのテーブル行に格納することが可能な新しいデータモデル。デバイスが複数の測定値を同時に送信した場合に、データ書き込み・保存・クエリスキャンのコスト効率に優れる。

time	deviceId	location	measure_name	temperature	humidity
2023-09-05 19:00:00	49820	東京	metrics	32.1	80
2023-09-05 19:00:01	1543	大阪	metrics	35.1	81
2023-09-05 19:00:02	59873	福岡	metrics	33.7	76

# 自動パーティショニング/インデクシングによるスケーリング



ディメンジョン

メジャー

time	deviceId	location	measure_name	temperature	humidity	...		
2023-09-05 19:00:00	49820	東京	metrics	32.1	80			
2023-09-05 19:00:01	1543	大阪	metrics	35.1	81			
2023-09-05 19:00:02	59873	福岡	metrics	33.7	76			
2023-09-05 19:00:03	6819	奈良	metrics	35.8	80			
2023-09-05 19:00:03	49	佐賀	metrics	31.5	67			
2023-09-05 19:00:03	9200	京都	metrics	36.9	81			
2023-09-05 19:00:03	1048	北海道	metrics	24.1	54			
2023-09-05 19:00:04	99	埼玉	metrics	34.3	79			
2023-09-05 19:00:04	439081	高知	metrics	36.4	75			
2023-09-05 19:00:05	4958	徳島	metrics	35.5	74			

- measure\_name が固定値だと、データが効率的に分散されない

# (元々の) ベストプラクティス

データを分散させる為に、dimension の値の Hash 値を 8,192 で割った余りを measure\_name に設定する事を推奨していた

time	deviceId	location	measure_name	temperature	humidity	...
2023-09-05 19:00:00	49820	東京	3948	32.1	80	
2023-09-05 19:00:01	1543	大阪	6781	35.1	81	
2023-09-05 19:00:02	59873	福岡	8100	33.7	76	
2023-09-05 19:00:03	6819	奈良	376	35.8	80	
2023-09-05 19:00:03	49	佐賀	221	31.5	67	
2023-09-05 19:00:03	9200	京都	7530	36.9	81	
2023-09-05 19:00:03	1048	北海道	5550	24.1	54	
2023-09-05 19:00:04	99	埼玉	6210	34.3	79	
2023-09-05 19:00:04	439081	高知	2321	36.4	75	
2023-09-05 19:00:05	4958	徳島	4432	35.5	74	

`hash( deviceId ) % 8192`

# ユーザ定義のパーティション機能が登場

任意の dimension 値にパーティションキーを設定できるようになった

time	deviceid	location	measure_name	temperature	humidity	...
2023-09-05 19:00:00	49820	東京	metrics	32.1	80	
2023-09-05 19:00:01	1543	大阪	metrics	35.1	81	
2023-09-05 19:00:02	59873	福岡	metrics	33.7	76	
2023-09-05 19:00:03	6819	奈良	metrics	35.8	80	
2023-09-05 19:00:03	49	佐賀	metrics	31.5	67	
2023-09-05 19:00:03	9200	京都	metrics	36.9	81	
2023-09-05 19:00:03	1048	北海道	metrics	24.1	54	
2023-09-05 19:00:04	99	埼玉	metrics	34.3	79	
2023-09-05 19:00:04	439081	高知	metrics	36.4	75	
2023-09-05 19:00:05	4958	徳島	metrics	35.5	74	

deviceid をパーティションキーに設定する事が出来る

# 4. ユーザ定義のパーティションキー

ディメンジョンをテーブルのパーティションのキーとして個別に設定できるようになり、設計時の選択肢が広がった

## スキーマの設定 - 新規 情報

スキーマは、テーブルに必要なデータモデルを指定します。パーティションキーを使用してデータを保存および配布する方法を指定します。カスタムパーティションキーを使用すると、クエリをより迅速かつ効率的に実行できます。

### パーティションキー設定

カスタムパーティションキーを作成するか、または Timestream のデフォルトパーティションキーを使用するかを選択します。

#### カスタムパーティショニング

これは、クエリの主なアクセスパターンが、カーディナリティの高いディメンジョンまたはメジャーをフィルタリングする場合に役立ちます。

#### デフォルトパーティショニング

これは、クエリの主なアクセスパターンが測定ベースのフィルタリングである場合に役立ちます。  
[measure\_name] 属性の値は、データをパーティショニングするためのパーティションキーとして使用されます。

## 複合パーティションキー 情報

テーブルデータのパーティショニングに使用される属性を定義するパーティションキーのリスト。リストの順序によってパーティショニング階層が決まります。

### 1-パーティションキータイプ

ディメンジョン ▼

### パーティションキー名

deviceid

#### レコードでパーティションキーを強制

新しいレコードを取り込む際には、ディメンジョンキーを指定する必要があります。

- ④ テーブルを作成した後に、パーティションキーを追加したり、削除したりすることはできません。キーは、テーブルが作成された後にテーブルの詳細ページで確認できます。

## 特徴

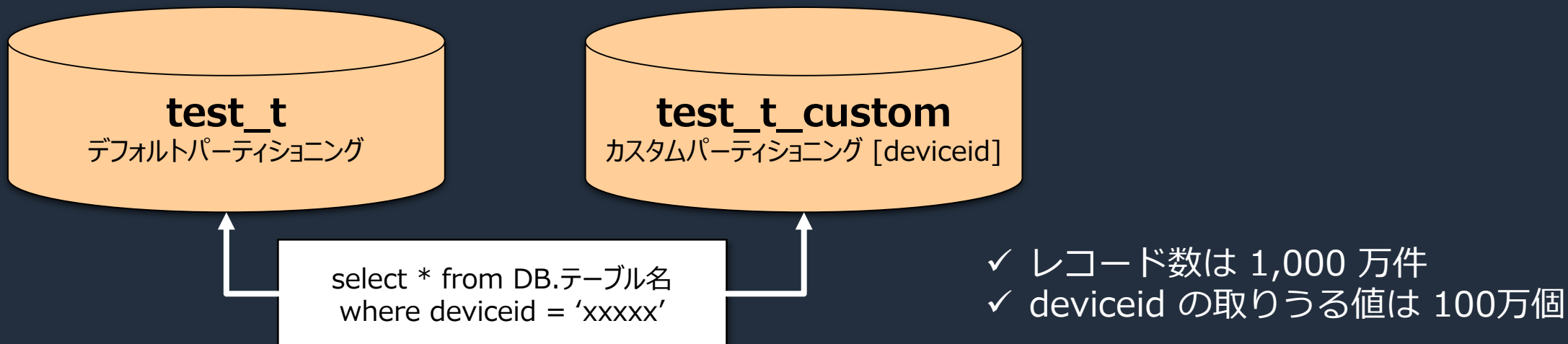
- 以下の条件に合致する場合、クエリのパフォーマンス改善に有用
  - デフォルトの `measure_name` でのパーティショニングでは目標のクエリパフォーマンスを達成出来ない
  - カーディナリティの高いディメンジョンがあり、かつ、改善が必要なクエリの `where` 条件に対象のディメンジョンが含まれる
- パーティションキーが必須かどうかをキー作成時に指定可能

## 注意点

- テーブル作成後にはパーティションキーを作成、変更、削除する事が出来ない為、入念な事前確認が必要

# デモの内容

- カスタムパーティションキーの設定したテーブルの作成
- SQL の実行
  - デフォルト/カスタムパーティショニングを設定したそれぞれのテーブルに対して同一 SQL を実行して性能差を確認





## テーブルを作成 情報

### テーブルの詳細

#### データベース名

このテーブルを作成するデータベースを選択します。

testdb



#### テーブル名

このデータベース内で一意のテーブル名を指定します。一度作成すると、この名前を変更することはできません。

myTable

3~256文字である必要があります。英字、数字、ダッシュ、ピリオド、またはアンダースコアを含める必要があります。

### スキーマの設定 - 新規 情報



# Amazon Timestream の日本語 Blog を提供しています

～ “aws blog timestream” で検索してみてください～

Amazon Web Services ブログ

Category: Amazon Timestream



**Amazon Timestream でスケジュールドクエリを使用して、クエリのパフォーマンスを向上させ、コストを削減する**  
by Tomohiro Miyazaki | on 10 8月 2023 | in Amazon Timestream, Best Practices, General, Intermediate (200), Technical How-To | [Permalink](#) | [Share](#)

本ブログでは、Amazon Timestream のスケジュールドクエリを使用してクエリのパフォーマンスを向 [...]



**Amazon Timestream で電力品質と顧客利用状況のデータを使って高調波の問題を分析、検出する方法**  
by Yosuke Nishihara | on 02 7月 2023 | in Advanced (300), Amazon Timestream, Best Practices, Technical How-To | [Permalink](#) | [Share](#)

本投稿では、Amazon Timestream とその組み込みの時系列の機能を利用して、数百万の顧客のメトリク [...]



**AWS AppSync で Amazon Timestream のデータにアクセス**  
by 稲田大陸 | on 09 6月 2023 | in Amazon Timestream, AWS AppSync, Front-End Web & Mobile | [Permalink](#) | [Share](#)

AWS AppSync は、複数のデータベース、マイクロサービス、API に安全にデータを検索または更新するための単一のエンドポイントを提供することによって、アプリケーション開発を簡素化するフルマネージドのサーバーレス GraphQL API サービスです。本記事では、エンドポイントディスカバリーパターンによる REST API を使用して、AWS AppSync 経由で Amazon Timestream データベースの時系列データにアクセスする方法を紹介します。



**Amazon Timestream でのマルチメジャーレコード、マグネティックストレージへの書き込み、スケジュールドクエリを利用した時系列データの格納と分析**  
by Yosuke Nishihara | on 31 5月 2023 | in Amazon Timestream | [Permalink](#) | [Share](#)

時系列は特にアプリケーションやインフラストラクチャー、IoT デバイス等から情報を収集する際の一般的なデータフ [...]



**Amazon Timestream のよくあるクエリパターンとその効率的な SQL 記述方法**  
by Yosuke Nishihara | on 17 5月 2023 | in Amazon Timestream | [Permalink](#) | [Share](#)

Amazon Timestream は時系列データを取り扱う為のサーバーレスの目的別データベースサービスであり [...]



Thank you!