JAPAN | JUNE 20, 2024

aws SUMMIT

マーケティングの岩田がこのページを作ります。後から展開します

マーケティングの岩田がこのページを作ります。後から展開します

Takashi Narita

Principal DynamoDB Specialist
Solutions Architect

アジェンダ

- 必ずSingle Table Designでなければいけないのか？
- 過去の経緯
- 現在の状況・どのような改善が行われたか
- サンプルで違いを確認する
- まとめ

必ずSingle Table Designでなければいけない？

たまに「DynamoDBを使うときはどんな時でも一つのテーブルにデータをまとめないといけないんですよね？」と聞かれる事があります。

結論から言うと必ずしも一つのテーブルにまとめる必要はありません。ケースに応じて有効なパターンとそうではないパターンがある。そのため、テーブル分割した方がメリットがある場合は積極的にテーブル分割をして欲しい。

AWS DB Blog : “Amazon DynamoDB におけるシングルテーブル vs マルチテーブル設計”

AWS Database Blog

Single-table vs. multi-table design in Amazon DynamoDB

by Alex DeBrie | on 16 AUG 2022 | in Amazon DynamoDB, Intermediate (200) | Permalink | Comments | Share

This is a guest post by Alex DeBrie, an AWS Hero.

For people learning about Amazon DynamoDB, the idea of single-table design is one of the most mind-bending concepts out there. Rather than the relational notion of having a table per entity, DynamoDB tables often include multiple different entities in a single table.

You can read the DynamoDB documentation, watch re:Invent talks or other videos, or check out my book to learn some of the design patterns with single-table design in DynamoDB. I want to look at the topic at a higher level with a specific focus on arguments both for and against single-table design.

In this post, we'll talk about single-table design in DynamoDB. We'll start off with some relevant background on DynamoDB that will inform the data modeling discussion. Then, we'll discuss when single-table design can be helpful in your application. Finally, we'll conclude with some instances where using multiple tables in DynamoDB might be better for you.

Relevant background on DynamoDB

翻訳版

https://aws.amazon.com/jp/blogs/news/single-table-vs-multi-table-design-in-amazon-dynamodb/

原文

https://aws.amazon.com/jp/blogs/database/single-table-vs-multi-table-design-in-amazon-dynamodb/

“Amazon DynamoDB におけるシングルテーブル vs マルチテーブル設計”より

シングルテーブルデザインの長所:

- 非正規化によりトランザクションを使わなくとも一貫性が保ちやすい
- 1つのテーブルから複数のビューを作成できる

シングルテーブルデザインの短所:

- データ冗長性が高くなる可能性がある
- クエリの複雑さが高くなる可能性がある

マルチテーブルデザインの長所:

- データの正規化が可能
- クエリが単純化される可能性がある

マルチテーブルデザインの短所:

- テーブル間のデータ取得について設計が複雑になる
- データ整合性を維持する必要がある

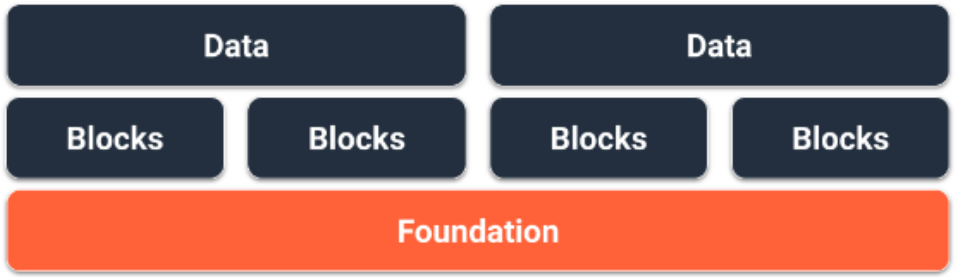
Developer Guide

DynamoDB のデータモデリングの基盤

Copy Plain Link | []l:t-s | []t-s-l

PDF | RSS

このセクションでは、基盤レイヤーについて、シングルテーブルとマルチテーブルという 2 種類のテーブル設計を検討します。



https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/data-modeling-foundations.html

過去の経緯など

過去なぜSingle Table Designがレコメンドされていたか

- “1アプリ“とマイクロサービスの粒度の認識相違
  - 「1アプリ1テーブル」の1アプリはマイクロサービスの1アプリ = 一つの役割を想定している事が多い。複雑な多くの機能を一つにまとめることをレコメンドしていた訳ではない
- 非正規化への意識の強調
  - RDBMSと同じテーブル設計にしてしまうことを少し強めのメッセージでマインドチェンジを促した？
- トランザクションの有無
  - 複数Item複数テーブルに分割した上で複数Itemの更新に一貫性が欲しかった場合実現が難しかった。シンプルな解決策として非正規化による単一アイテム操作に変える事で Atomic処理を実現出来るように促す必要があった

過去の経緯など

過去なぜSingle Table Designをレコメンドしていたのか？

管理上の側面

- パーティションあたりのキャパシティ希薄化の挙動
  - Adaptive Capacityが無い頃、テーブルキャパシティを下げた場合パーティションあたりの希薄化を防ぐためにある程度のキャパシティを常に維持した方がスロットリング回避に有利だった
- On-demand modeが無い時期のCapacity管理
  - Auto Scalingを組み合わせたとしても、ほぼアクセスが無いテーブルへもサービスダウンしないように最小スループットを設定しておく必要があり、大量のテーブルで無駄なキャパシティが生まれる場合、1テーブルにマージした方が非常に効率的だった
- Table数の上限
  - 以前はマルチテナント型設計の場合テーブル数の引き上げにも限界があった。 1アカウントあたりの上限引き上げ(2022年)があり多くの場面で解決済み。
- メトリクスなどを過剰に増やさないようにすることでのcost saving

  テーブル単位で生成されるメトリクス、それらを監視するツールなどの集約

現時点での推奨は？

Multi Tableの方が良い場面

- テーブルレベルでIAMを明確に分けたい
- ユーザープロフィールとHistoryデータなどデータの増加特性などが大きく異なるもの
- 肥大化するものが同居すると、バックアップやOLAPワークロードなどでデータの分離をどうするか？という課題が出てくる
- 月単位でDELETE Tableが出来たら運用上大きくオペレーションコストが下がるような時系列データ

現時点での推奨は？

Single Tableの方が良い場面

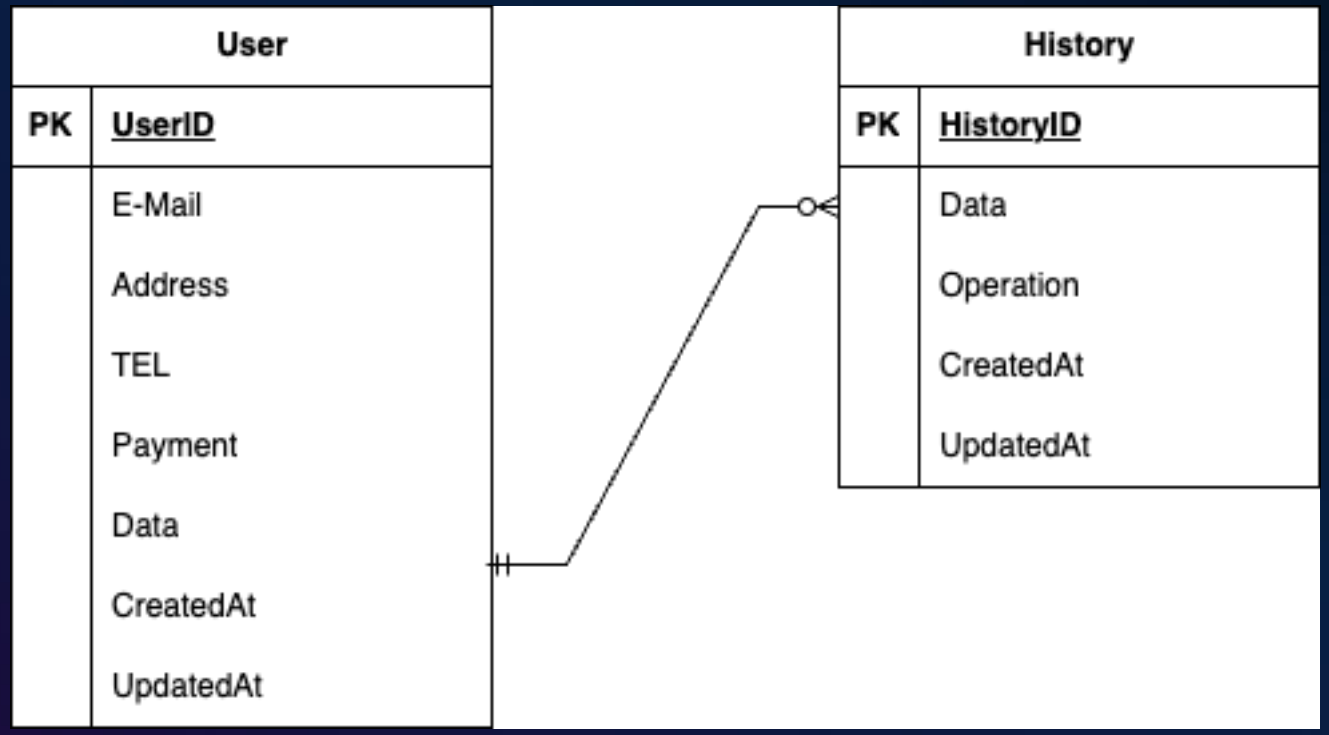
- 複数テーブルにすることでテーブル数の増加によるコスト増、オペレーション増加が見えている場合 -> 1TableならQueryで済むのに複数テーブルに分かれる事で大量のGetItemになる、など
- マルチテナント方式で各テナントの少数のデータをそれぞれ別のテーブルにしていた、などの同一コンテキスト別テーブルのような場合、CloudWatchMetricsや3rd Partyの監視などtable数で肥大化してしまうようなものの節約
- バックアップ数の集約など効率化が見込める

現時点での推奨は？

- Single/Multiはどちらが正解、ではなく自分たちのワークロードがあった上でどちらのほうが最適化できているか？
- 重要なのはSingle -> Multi, Multi -> Singleどちらのパターンでも自身のワークロードを見極め、必要に応じて変化させる。どうやってマイグレーションをするか？ということを想定する
- どのように既存テーブルのデータを別テーブルに分割・統合するか？
- DynamoDBではDynamoDB Streams, Export to S3(Full, 差分)など複数の手法が取れるので必要に応じてプロセスを設計

簡単なサンプルで比較

1. ユーザープロフィールと履歴情報でSingle or Multi
2. ユーザープロフィールをSingle 1 Item or 2 Item or Multi



Single Table(User + History)

Primary key		Attributes				
Partition key: PK	Sort key: SK					
User1	History_2024-05-28T03:02:11Z	Data	Operation	CreatedAt	UpdatedAt	
		qwertyuiop	operation_1	2024-05-28T03:02:11Z	2024-05-28T03:02:11Z	
	History_2024-05-28T03:02:26Z	Data	Operation	CreatedAt	UpdatedAt	
		作業データ:aaaaaaa	operation_5	2024-05-28T03:02:26Z	2024-05-28T03:02:26Z	
	Profile	E-Mail	Address	TEL	Payment	Data
		user1@example.com	東京都品川区上大崎1--1	03-1234-5678	CreditCard_A	aaabbbccc

Multi Table User and History

User

Primary key		Attributes					
Partition key: PK	Sort key: SK						
		E-Mail	Address	TEL	Payment	CreatedAt	UpdatedAt
User1	All	user2@example.com	大阪府大阪市どこか1-1	06-9876-5432	Credit_ZZZZ	2024-05-28T04:14:07Z	2024-05-28T04:14:07Z

History

Primary key		Attributes			
Partition key: PK	Sort key: SK				
		Data	Operation	CreatedAt	UpdatedAt
User1	2024-05-28T03:50:10Z	aaabbbccc	Operation_1	2024-05-28T03:50:10Z	2024-05-28T03:50:10Z
		Data	Operation	CreatedAt	UpdatedAt
	2024-05-28T03:50:50Z	qwertyuiop	Operation_3	2024-05-28T03:50:50Z	2024-05-28T03:50:50Z
		Data	Operation	CreatedAt	UpdatedAt
	2024-05-28T03:50:56Z	登録データ:AAA	Operation_27	2024-05-28T03:50:56Z	2024-05-28T03:50:56Z

User : Single Item vs Multi Item

Single Item

Primary key		Attributes						
Partition key: PK	Sort key: SK							
User1	All	E-Mail	Address	TEL	Payment	Data	CreatedAt	UpdatedAt
		user1@example.com	大阪府大阪市どこか1-1	06-9876-5432	Credit_ZZZZ	LargeData	2024-05-28T04:14:07Z	2024-05-28T04:14:07Z

Item一つでユーザー情報を表現可能

Multi Item

User2	Payment	Payment	Data	CreatedAt	UpdatedAt	
		Credit_CCCCC	LargeData	2024-06-12T02:02:26Z	2024-06-12T02:02:26Z	
	Profile	E-Mail	Address	TEL	CreatedAt	UpdatedAt
		user2@example.com	北海道札幌市中央区3-3-3	011-123-4567	2024-06-12T02:02:26Z	2024-06-12T02:02:26Z

2つ合わせてユーザー情報、片方だけでもQueryでまとめて取得のどちらも可能

User : Single Item vs Multi Table

Multi Table/Multi Item

User3	Profile	E-Mail	Address	TEL	CreatedAt	UpdatedAt
		user3@example.com	東京都世田谷区1-1-1	03-1234-5678	2024-06-12T04:14:24Z	2024-06-12T04:14:24Z

Primary key		Attributes			
Partition key: PK	Sort key: SK				
User3	Payment	Payment	Data	CreatedAt	UpdatedAt
		Credit_AAAA	LargeData	2024-06-12T02:00:46Z	2024-06-12T02:00:46Z

2つ合わせてユーザー情報になる。テーブルが違うのでテーブルレベルの管理を分割出来るが個別にGetItemで取得する必要がある

パターンのまとめ

- Single Table & Single Item
  - Pros : 一度の書き込み、読み込みで必要なデータが全て取得可能
  - Cons :LargeDataを含んでいるとLargeDataが不要な場合でも操作が高コスト
- Single Table & Multi Item
  - Pros : Queryで選択的に取得も可能、LargeDataを含むItemも分割する事でコストのコントロールもしやすい
  - Cons : 書き込み時に分割したItem分のコスト増になる可能性がある
- Multi Item & Multi Table
  - Pros : Multi Itemを更に別テーブルにする事でIAMやバックアップなどの個別コントロールも行える
  - Cons : Queryでまとめて取得が出来ないためにMulti Itemより更にデータ操作回数が増える可能性がある

まとめ

DynamoDB ではとにかくSingle Tableにこだわらず、Single Table/Multi Tableを使い分けるのが良い

それぞれのメリット・デメリットを理解するのが重要

もし判断に迷ったらそれぞれをモデリングしてみて、データ操作、運用面で個別にメリット・デメリットを整理・比較してみる

ワークロードや要件が変わった場合には比較し直してみて、どうすれば今のモデリングで抱えている問題が解決するか、どうやって新しいモデリングに移行するかを考えておく

Thank you!