



# Amazon DynamoDB

## Database Fundamentals

© 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

Amazon DynamoDB 를 시작하겠습니다.

## 교육 내용

### 이 강의의 핵심

배울 내용은 다음과 같습니다.

- Amazon DynamoDB 데이터베이스 서비스의 세부 정보 설명
- 관계형 데이터베이스와 비관계형 데이터베이스의 차이 조명

주요 용어:

- Amazon DynamoDB
- 파티션
- 관계형 데이터베이스
- 비관계형 데이터베이스



이 모듈에서는 데이터베이스 솔루션과 관련된 주요 개념을 학습합니다.

배울 내용은 다음과 같습니다.

- Amazon DynamoDB 데이터베이스 서비스의 세부 정보 설명
- 관계형 데이터베이스와 비관계형 데이터베이스의 차이 조명

이 모듈의 목표는 솔루션을 강화하는 데 사용할 수 있는 데이터베이스 리소스를 이해하는 데 도움을 주기 위한 것입니다. 또한 다양한 선택 사항이 솔루션 가용성에 미치는 영향을 이해할 수 있도록 사용 가능한 다양한 서비스 기능을 검토합니다.



## Amazon DynamoDB

NoSQL AWS 데이터베이스 서비스인 Amazon DynamoDB 소개를 시작하겠습니다.

**Amazon DynamoDB**는 완전관리형 NoSQL 데이터베이스 서비스입니다. Amazon은 이 서비스의 모든 기반 데이터 인프라를 관리합니다. 리전에서 내결함성 아키텍처의 일부로 여러 시설에 중복으로 데이터를 저장합니다. DynamoDB를 사용하면 테이블과 항목을 생성할 수 있습니다. 테이블에 항목을 추가할 수 있습니다. 자동으로 데이터를 분할하며 테이블 스토리지를 제공하여 워크로드 요구 사항을 충족합니다.

DynamoDB는 완전관리형 NoSQL 데이터베이스 서비스입니다. Amazon은 이 서비스의 모든 기반 데이터 인프라를 관리합니다. 리전에서 내결함성 아키텍처의 일부로 여러 시설에 중복으로 데이터를 저장합니다. DynamoDB를 사용하면 테이블과 항목을 생성할 수 있습니다. 테이블에 항목을 추가할 수 있습니다. 자동으로 데이터를 분할하며 테이블 스토리지를 제공하여 워크로드 요구 사항을 충족합니다.

## Amazon DynamoDB 기술 이점

**Amazon DynamoDB는 빠르고 확장성이 우수한 비관계형 데이터베이스 서비스입니다.**

Amazon DynamoDB의 이점은 다음과 같습니다.

- 완전관리형
- 대기 시간이 짧은 쿼리
- 세분화된 액세스 제어
- 유연성



Amazon DynamoDB는 문서 모델과 키 값 스토어 모델을 모두 지원하는 빠르고 유연한 NoSQL 데이터베이스 서비스입니다. 이점은 다음과 같습니다.

- **완전관리형 서비스** - 데이터베이스 테이블을 생성하고 오토스케일링을 위한 목표 처리량을 설정하면 서비스가 자동으로 데이터베이스 관리 작업을 수행합니다. 하드웨어나 소프트웨어 프로비저닝, 설정 및 구성, 소프트웨어 패치, 분산형 데이터베이스 클러스터 운영, 크기 조정에 따른 여러 인스턴스의 데이터 파티셔닝을 처리합니다. 또한 DynamoDB는 모든 테이블에 대해 시점 복구, 백업 및 복원을 제공합니다.
- **대기 시간이 짧은 쿼리** - 쿼리를 실행할 때 평균 서비스 측 대기 시간은 일반적으로 10밀리초 미만입니다. 데이터 볼륨이 증가하고 애플리케이션 성능 요구가 늘어남에 따라 DynamoDB는 이러한 요구에 맞게 조정됩니다. 자동 파티셔닝 및 SSD 기술을 사용하여 처리량 요구 사항을 충족하고 규모와 관계없이 짧은 대기 시간을 제공합니다.
- **세분화된 액세스 제어** - DynamoDB를 AWS Identity 및 Access Management(IAM)와 함께 사용하여 조직 내 사용자의 액세스를 세부적으로 제어할 수 있습니다. 따라서 개별 사용자에게 고유한 보안 인증을 할당하고 서비스 및 리소스에 대한 각 사용자의 액세스를 제어할 수 있습니다.
- **유연성**: DynamoDB는 데이터를 JSON(JavaScript Object Notation) 문서로 저장, 쿼리 및 업데이트하는 것을 지원합니다. 이러한 지원으로 인해 반정

형 데이터를 저장하고 JSON 쿼리를 사용하여 이를 조작하기에 적합합니다

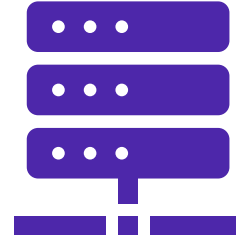
.

또한 Amazon DynamoDB를 Amazon CloudWatch와 함께 사용하여 처리량과 대기 시간 요청 통계를 확인할 수 있습니다.

# Amazon DynamoDB 이해

## Amazon DynamoDB 이해

- 일부 Amazon DynamoDB 고객은 Amazon DynamoDB에 수십억 개의 항목을 포함하는 테이블을 보유하고 있습니다.
- 새로운 형식의 항목은 스키마 마이그레이션을 수행할 필요 없이 동일한 테이블에 이전 항목과 나란히 저장할 수 있습니다.
- 애플리케이션이 점점 더 많이 사용되고 사용자가 이 애플리케이션과 계속 상호 작용할 경우 애플리케이션의 요구에 따라 데이터베이스가 증가할 수 있습니다.



DynamoDB를 사용하면 테이블에 저장할 수 있는 항목 수는 실제로 제한이 없습니다. 예를 들어 일부 고객은 프로덕션 테이블에 수십억 개의 항목이 포함되어 있습니다.

NoSQL 데이터베이스의 이점 중 하나는 동일한 테이블의 항목이 다른 속성을 가질 수 있다는 점입니다. 따라서 애플리케이션이 증가할 때 속성을 추가할 수 있는 유연성이 제공됩니다. 새로운 형식의 항목은 스키마 마이그레이션을 수행할 필요 없이 동일한 테이블에 이전 항목과 나란히 저장할 수 있습니다.

애플리케이션이 더 많이 사용되고 사용자가 이 애플리케이션과 계속 상호 작용할 경우 애플리케이션의 요구에 따라 스토리지를 확장할 수 있습니다.

DynamoDB의 모든 데이터는 SSD에 저장되며, 간단한 쿼리 언어를 사용하여 대기 시간이 짧은 쿼리 성능을 일관되게 유지할 수 있습니다.

스토리지 확장 외에 DynamoDB는 테이블에 필요한 읽기 또는 쓰기 처리량도 프로비저닝할 수 있습니다. 애플리케이션 사용자 수가 증가하면, 수동 프로비저닝을 통해 증가된 수의 읽기/쓰기 요청을 처리할 수 있도록 DynamoDB 테이블의 크기를 조정할 수 있습니다. 또는 DynamoDB가 테이블의 로드를 모니터링하고 자동으로 프로비저닝된 처리량을 늘리거나 줄일 수 있도록 오토스케일링을 활

성화할 수 있습니다.

추가적인 주요 차별화 기능으로는 선택한 AWS 리전에 걸친 자동 복제를 지원하는 전역 테이블, 저장 데이터의 암호화 및 항목 유지 시간(TTL)이 있습니다.





핵심 개념: 테이블, 항목 및 속성

Amazon DynamoDB는 NoSQL 데이터베이스 서비스입니다.

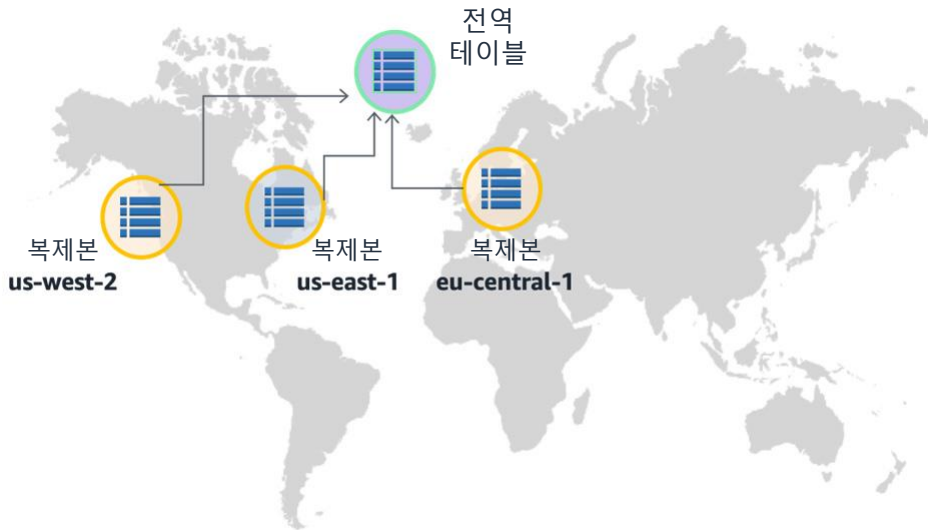
# 주요 개념

## DynamoDB의 핵심 개념:

- **테이블** - 다른 데이터베이스 시스템과 유사하게 DynamoDB는 테이블에 데이터를 저장합니다.  
**테이블**은 데이터의 모음입니다.
  - 예를 들어 친구, 가족 또는 기타 관심 있는 사람에 대한 개인 연락처 정보를 저장하는 데 사용할 수 있는 **People**이라는 테이블이 있을 수 있습니다.
- **속성** - 각 항목은 하나 이상의 속성으로 구성됩니다.  
**속성**은 더 이상 나눌 필요가 없는 기본적인 데이터 요소입니다.
  - 예를 들어 **People** 테이블의 항목에는 **PersonID**, **LastName**, **FirstName** 등의 속성이 있을 수 있습니다.
- **항목** - 각 테이블에는 0개 이상의 항목이 있습니다.  
**항목**은 전체 항목 중에서 고유하게 식별할 수 있는 속성의 그룹입니다.
  - 예를 들어 예시의 **People** 테이블에서 각 항목은 사람을 나타냅니다.

```
{  
  "PersonID": 101,  
  "LastName": "Smith",  
  "FirstName": "Fred",  
  "Phone": "555-4321"  
}
```

## Amazon DynamoDB 전역 테이블



9

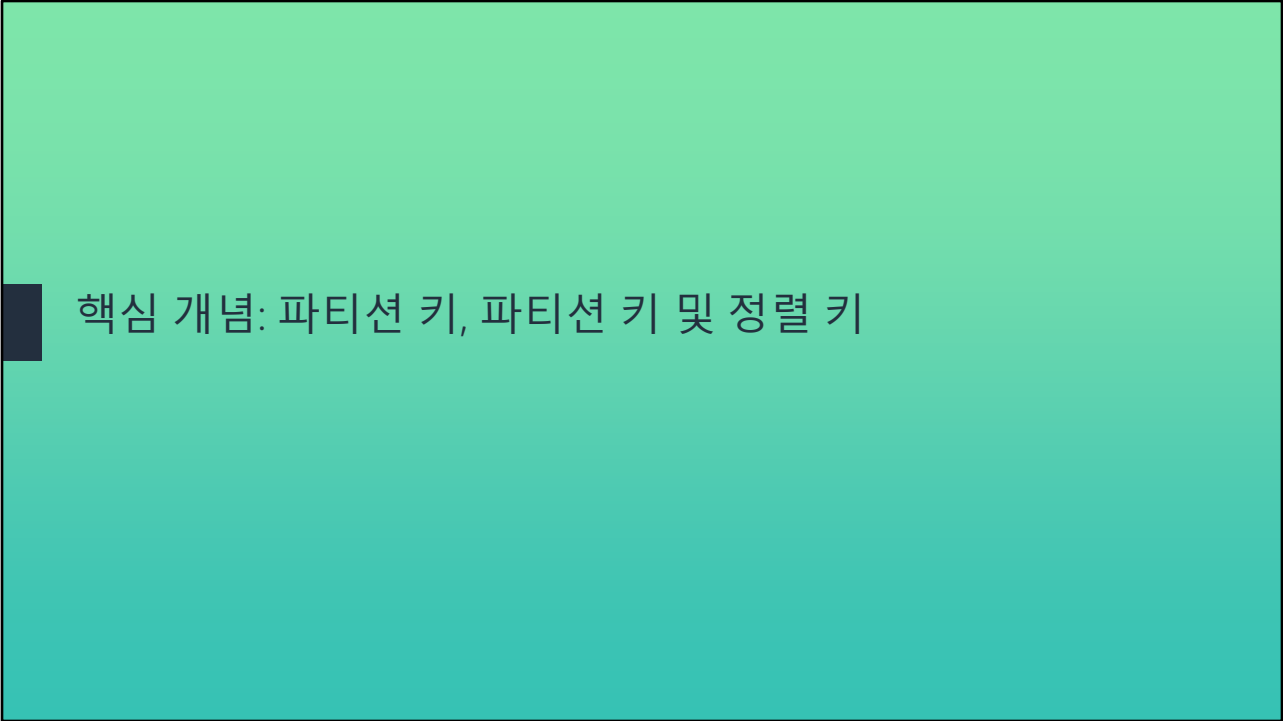
aws re/start

DynamoDB 전역 테이블 기능은 여러 리전 간에서 고가용성과 확장성을 제공합니다.

**전역 테이블**은 단일 AWS 계정이 소유해야 하는 한 개 이상의 DynamoDB 테이블의 모음입니다. 모음의 테이블은 복제 테이블이라고도 합니다. **복제 테이블**(또는 **복제본**)은 전역 테이블의 일부로 기능하는 단일 DynamoDB 테이블입니다. 각 복제본에는 동일한 데이터 항목 집합이 저장됩니다. 전역 테이블은 AWS 리전당 한 개의 복제 테이블만 가질 수 있습니다. 모든 복제본은 동일한 테이블 이름과 동일한 기본 키 정의를 가집니다.

전역 테이블을 만들 때, 테이블을 제공하고자 하는 AWS 리전을 지정해야 합니다. DynamoDB는 이러한 리전에 동일한 테이블을 만들고, 이 모든 테이블에 대한 지속적인 데이터 변경을 전파하기 위해 필요한 모든 태스크를 수행합니다.

DynamoDB 전역 테이블은 전역적으로 분산된 사용자가 있는 대규모 애플리케이션과 함께 잘 작동합니다. 이러한 환경에서 사용자는 자신과 근접한 복제본에 액세스하여 빠른 애플리케이션 성능을 기대할 수 있습니다. 또한, AWS 리전 중 하나를 임시로 사용하지 못하게 되더라도 사용자는 여전히 다른 리전에서 동일한 데이터에 액세스할 수 있습니다.



핵심 개념: 파티션 키, 파티션 키 및 정렬 키

Amazon DynamoDB는 NoSQL 데이터베이스 서비스입니다.

# 테이블 및 데이터

## 핵심 구성 요소

- DynamoDB는 2가지 종류의 기본 키를 지원합니다.
  - 파티션 키
  - 파티션 및 정렬 키



DynamoDB는 2가지 종류의 기본 키를 지원합니다.

## 키 이해

테이블을 생성할 때 테이블 이름과 더불어 테이블의 기본 키를 지정해야 합니다.

- 파티션 키는 파티션 키라는 하나의 속성으로 구성된 단순 기본 키입니다.
- 파티션 키와 정렬 키는 2개의 속성으로 구성된 복합 기본 키라고도 합니다.



1개의 속성



2개의 속성

**기억할 사항:** 기본 키는 테이블의 각 항목을 고유하게 식별하므로 두 항목이 동일한 키를 가질 수 없습니다.

=**파티션 키**는 **파티션 키**라는 하나의 속성으로 구성된 단순 기본 키입니다. 파티션 키와 정렬 키는 2개의 속성으로 구성된 **복합 기본 키**라고도 합니다. 복합 키가 무엇인지 더 잘 이해하려면 다음 예시를 살펴 보십시오. 테이블, 항목 및 속성에 설명한 Music 테이블은 복합 기본 키(Artist 및 SongTitle)가 있는 테이블의 예시입니다.

DynamoDB의 작동 방식에 대해 자세히 알아보려면 [Amazon DynamoDB 핵심 구성 요소](#)를 참조하십시오.

## 음악 테이블 예시

다이어그램은 몇 가지 예시 항목과 속성 및 기본 키가 있는 **Music**이라는 테이블을 보여줍니다.

- Music의 **기본 키**는 2개의 속성으로 구성됩니다 (**Artist** 및 **SongTitle**).
  - 테이블의 각 항목은 이러한 두 속성을 가져야 합니다.
- **Music** 테이블은 **스키마가 없으며**, 이는 속성이나 데이터 유형을 사전에 정의할 필요가 없음을 의미합니다. 각 항목은 자체의 개별 속성을 가질 수 있습니다.

### Music

```
{
  "Artist": "No One You Know",
  "SongTitle": "My Dog Spot",
  "AlbumTitle": "Hey Now",
  "Price": 1.98,
  "Genre": "Country",
  "CriticRating": 8.4
}
```

```
{
  "Artist": "No One You Know",
  "SongTitle": "Somewhere Down The Road",
  "AlbumTitle": "Somewhat Famous",
  "Genre": "Country",
  "CriticRating": 8.4,
  "Year": 1984
}
```

Music이라는 이름의 테이블을 생각해 보십시오. 테이블에는 2개의 속성(**Artist**와 **SongTitle**)으로 구성된 기본 키가 있습니다. 테이블의 각 항목은 이러한 두 속성을 가져야 합니다. **Artist**와 **SongTitle**의 조합은 테이블의 각 항목을 다른 항목과 구분합니다.

기본 키 이외에, DynamoDB에서 **Music** 테이블은 스키마가 없으며, 이는 속성이나 데이터 유형을 사전에 정의할 필요가 없음을 의미합니다. 각 항목은 자체의 개별 속성을 가질 수 있습니다.



## 작동 방식

Amazon DynamoDB의 작동 방식을 살펴보겠습니다.



# Amazon DynamoDB가 데이터를 분산하는 방식

## 데이터 분산 방식 이해

### DynamoDB가 데이터를 저장하는 방법:

- Amazon DynamoDB는 파티션에 데이터를 저장합니다. **파티션**은 SSD(Solid State Drive)로 지원되는 테이블용 스토리지 할당으로, 한 AWS 리전 내의 여러 가용 영역에 자동으로 복제됩니다.

### DynamoDB가 항목을 저장하고 검색하는 방법:

- 테이블에 기본 키(파티션 키만)가 있는 경우, DynamoDB는 해당 파티션 키 값에 따라 각 항목을 **저장하고 검색**합니다.

### DynamoDB가 테이블에 항목을 쓰는 방법:

- DynamoDB는 내부 해시 함수에 대한 입력으로 파티션 키의 값을 사용합니다. 해시 함수 출력 값에 따라 항목을 저장할 파티션이 결정됩니다.

### DynamoDB가 테이블에서 항목을 읽는 방법:

- 테이블에서 항목을 읽으려면 항목의 파티션 키 값을 지정해야 합니다. DynamoDB는 이 값을 해시 함수에 대한 입력으로 사용하여 항목을 찾을 수 있는 파티션을 산출합니다.

Amazon DynamoDB가 데이터를 분산하는 방법을 이해하기 위해 데이터를 저장하고, 데이터를 테이블에 쓰고, 테이블에서 검색하는 방법을 살펴보겠습니다.

## DynamoDB가 데이터를 저장하는 방법:

Amazon DynamoDB는 파티션에 데이터를 저장합니다. **파티션**은 SSD(Solid State Drive)로 지원되는 테이블용 스토리지 할당으로, 한 AWS 리전 내의 여러 가용 영역에 자동으로 복제됩니다.

## DynamoDB가 항목을 저장하고 검색하는 방법:

테이블에 기본 키(파티션 키만)가 있는 경우, DynamoDB는 해당 파티션 키 값에 따라 각 항목을 **저장하고 검색**합니다.

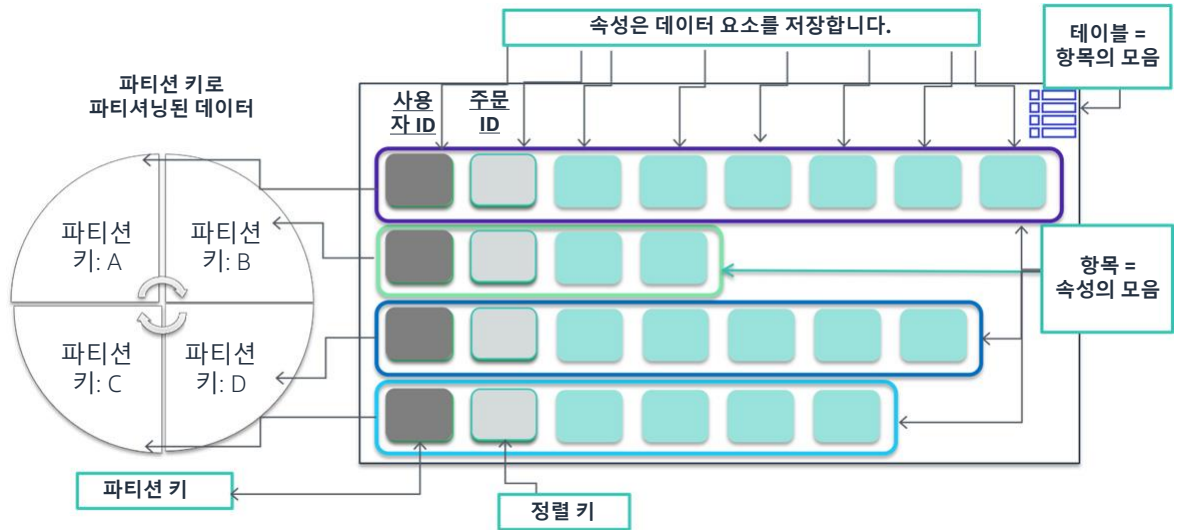
## DynamoDB가 테이블에 항목을 쓰는 방법:

DynamoDB는 내부 해시 함수에 대한 입력으로 파티션 키의 값을 사용합니다. 해시 함수 출력 값에 따라 항목을 저장할 파티션이 결정됩니다.

## **DynamoDB가 테이블에서 항목을 읽는 방법:**

테이블에서 항목을 읽으려면 항목의 파티션 키 값을 지정해야 합니다. DynamoDB는 이 값을 해시 함수에 대한 입력으로 사용하여 항목을 찾을 수 있는 파티션을 산출합니다.

## Amazon DynamoDB: 작동 방식



16

aws re/start

DynamoDB에서는 데이터가 **테이블**에 저장됩니다. 테이블에는 **속성**이 있는 **항목**이 포함되어 있습니다. 항목은 관계형 데이터베이스의 행 또는 튜플로, 속성은 열로 간주할 수 있습니다.

DynamoDB는 데이터를 파티션에 저장하고 테이블의 항목을 **파티션 키** 값에 따라 여러 파티션으로 나눕니다. **파티션**은 테이블의 스토리지 할당입니다. SSD로 지원되며 한 AWS 리전의 여러 가용 영역에 걸쳐 자동으로 복제됩니다.

DynamoDB는 파티션 관리를 처리합니다. 항목의 파티션 키를 **해시 속성**이라고도 합니다.

**정렬 키**는 파티션 키 값이 동일한 모든 항목을 서로 물리적으로 가까이 저장하도록 정의할 수 있습니다. 정렬 키는 파티션의 정렬 키 값으로 항목의 순서를 지정할 수 있습니다. 파티션 키를 기준으로 일대다 관계를 나타내며 정렬 키 속성에 대한 쿼리를 지원합니다.

테이블에는 테이블의 각 항목을 고유하게 식별하는 기본 키가 있습니다. 기본 키의 2가지 유형은 다음과 같습니다.

- **파티션 기본 키** - 기본 키는 단일 속성인 파티션 키로 구성됩니다. DynamoDB는 이 기본 키 속성에 순서가 지정되지 않은 인덱스를 생성합니다. 테이블의 각 항목은 파티션 키 값에 따라 고유하게 식별됩니다.

- **파티션 및 정렬 기본 키** - 기본 키는 2가지 속성으로 구성됩니다. 첫 번째 속성은 파티션 키 속성이고, 두 번째 속성은 정렬 키 속성입니다. DynamoDB는 파티션 키 속성에 순서가 지정되지 않은 인덱스를, 정렬 키 속성에 순서가 지정된 인덱스를 생성합니다. 테이블의 각 항목은 파티션 키 값과 정렬 키 값의 조합으로 고유하게 식별됩니다.

예시의 테이블에는 **User ID**와 **Order ID** 속성으로 이루어진 파티션 및 정렬 기본 키가 있습니다. DynamoDB 데이터 모델 및 테이블에 대한 자세한 내용은 [Amazon DynamoDB란?](#)을 참조하십시오.





## 파티셔닝의 개념

파티셔닝의 작동 방식을 살펴보겠습니다.

## 파티셔닝과 파티셔닝의 개념



 데이터는 테이블에 저장됩니다.

 테이블 데이터는 기본 키를 기준으로 파티셔닝되고 인덱싱됩니다.



쿼리는 기본 키를 사용하여 효과적으로 항목을 찾기 위해 테이블 파티셔닝을 사용합니다.

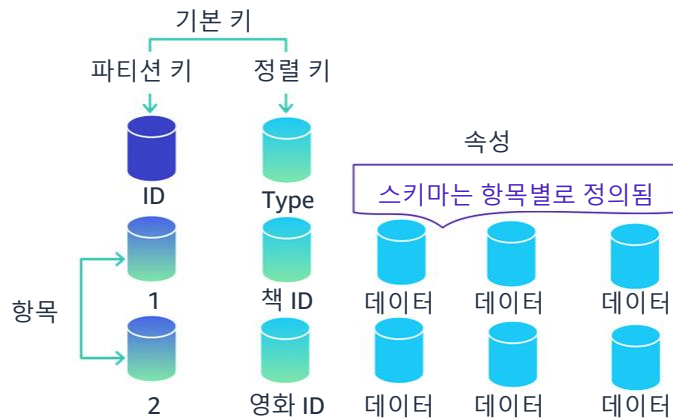
데이터가 증가함에 따라 테이블 데이터는 기본 키를 기준으로 파티셔닝되고 인덱싱됩니다.

DynamoDB 테이블에서 데이터를 검색하는 방법은 2가지가 있습니다.

- 첫 번째 방법에서는 쿼리 **작업**이 기본 키를 사용하여 항목을 효과적으로 찾기 위해 테이블 파티셔닝을 활용합니다.
- 두 번째 방법은 **스캔**을 사용하며, 키가 아닌 속성의 조건을 매칭하여 테이블의 항목을 찾을 수 있습니다.
  - 두 번째 방법은 다른 속성을 기준으로 항목을 찾을 수 있는 유연성을 제공합니다.
  - 그러나 DynamoDB가 테이블의 모든 항목을 스캔하여 기준에 맞는 항목을 찾기 때문에 작업의 효율성이 떨어집니다.

# 파티셔닝

## 파티셔닝의 개념



**파티션**은 SSD(Solid State Drive)로 지원되는 테이블용 스토리지 할당으로, 한 AWS 리전 내의 여러 가용 영역에 자동으로 복제됩니다. 자세한 내용은 [파티션 및 데이터 분산](#) 리소스를 참조하십시오.

## Amazon DynamoDB 데모

DynamoDB 데모: Amazon DynamoDB 콘솔 데모를 검토하십시오.  
이 동영상 데모는 학습 관리 시스템에서 찾을 수 있습니다.



## 학습 내용 확인 질문



프로젝트를 신속하게 시작해야 할 때 비관계형(NoSQL) 데이터베이스가 좋은 선택인 이유는 무엇입니까?



데이터가 파티셔닝되는 위치를 담당하는 요소는 무엇입니까?



Amazon DynamoDB 테이블에서 항목을 생성할 때 각 항목은 동일한 속성을 가져야 합니까?

정답:

1. NoSQL 데이터베이스에는 유연한 스키마가 있어 데이터 엔지니어링이 필요하지 않습니다. 개발자는 데이터 간 관계를 이해할 필요 없이 신속하고 간편하게 데이터로 작업할 수 있습니다.
2. 데이터가 파티셔닝되는 위치를 담당하는 것은 기본 키입니다.
3. 아니요. DynamoDB는 NoSQL 데이터베이스이기 때문에 개별 항목은 다른 항목과 다른 속성을 가질 수 있습니다.

## 핵심 사항



© 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

22

- DynamoDB는 완전관리형 NoSQL 데이터베이스 서비스입니다.
- DynamoDB는 모든 규모에서 10밀리초 미만의 일관된 대기 시간을 제공합니다.
- DynamoDB에는 사실상 테이블 크기 및 처리량 제한이 없습니다.
- 전역 테이블을 사용하면 리전 간에 데이터를 복제하고 업데이트 충돌을 해결해야 하는 문제를 없애줍니다.

aws re/start

이 강의에서 다룬 핵심 사항은 다음과 같습니다.

- DynamoDB는 완전관리형 NoSQL 데이터베이스 서비스입니다.
- DynamoDB는 모든 규모에서 10밀리초 미만의 일관된 대기 시간을 제공합니다.
- DynamoDB에는 사실상 테이블 크기 및 처리량 제한이 없습니다.
- 전역 테이블을 사용하면 리전 간에 데이터를 복제하고 업데이트 충돌을 해결해야 하는 문제를 없애줍니다.