**실습 3 (Java) - Amazon DynamoDB를 사용한 솔루션 개발**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. 본 내용은 Amazon Web Services, Inc.의 사전 서면 허가 없이 전체 또는 일부를 복제하거나 재배포할 수 없습니다. 상업적인 복제, 임대 또는 판매는 금지됩니다. 모든 상표는 해당 소유자의 자산입니다.

참고: 개인 정보, 개인을 식별할 수 있는 정보 또는 기밀 정보는 실습 환경에 입력하지 마십시오. 입력한 정보가 다른 사용자에게 공개될 수도 있습니다.

수정 사항이나 피드백 또는 기타 질문이 있으십니까? [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training)에서 문의해 주십시오.

**소요 시간**

이 실습은 완료하는 데 약 **60분**이 소요됩니다.

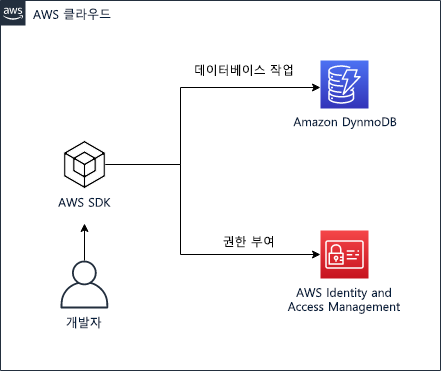
**선행 조건**

이 실습에는 다음이 필요합니다.

* Wi-Fi가 연결된 Microsoft Windows 또는 macOS 노트북 컴퓨터.
* 인터넷 브라우저(예: Chrome, Firefox 또는 IE9 이상).
* **참고:** 이전 버전의 Internet Explorer는 지원되지 않습니다.
* 개발 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 연결하려면 Microsoft 원격 데스크톱 클라이언트가 필요합니다.
* **참고:** 이 지침은 iPad 또는 태블릿 디바이스를 사용해 실습 콘솔에서 확인할 수 있습니다.
* **추가 정보:** 추가 실습 환경 관련 세부 정보는 [부록](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#appendix)을 참조하십시오.

**개요**

이 실습에서는 AWS CLI를 사용하여 노트를 저장할 **Notes**라는 Amazon DynamoDB 테이블을 생성하고 이 테이블의 데이터를 처리합니다. 필요에 따라 테이블의 키 속성을 정의합니다. 이전 실습에서 생성한 JSON 파일에서 샘플 테스트 데이터를 로드하고 다양한 쿼리를 작성하여 애플리케이션에 사용된 특정 노트를 검색합니다. 속성 조건에 따라 특정 노트를 업데이트합니다.



처음에는 불완전한 소스 코드가 제공됩니다. 주어진 단서를 사용하여 TODO 자리 표시자에서 코드의 누락된 부분을 완성합니다. 구문 오류가 없는 코드를 채운 다음 실행하여 결과를 검토합니다.

**목표**

이 실습을 완료하면 다음을 할 수 있게 됩니다.

* 프로그램의 하위 수준, 문서 및 상위 수준 API를 사용하여 프로그래밍 방식으로 DynamoDB와 상호 작용합니다.
* 파티션 키, 정렬 키 및 적합한 프로비저닝된 처리량을 포함한 Waiter를 사용하여 테이블을 생성합니다.
* 파일에서 JSON 객체를 읽고 테이블을 로드합니다.
* 키 속성, 필터, 표현식 및 페이지 배열을 기반으로 테이블에서 항목을 검색합니다.
* 새 속성을 추가하고 조건부로 데이터를 변경하여 항목을 업데이트합니다.
* 객체 지속성 모델을 사용하여 DynamoDB 데이터에 액세스합니다.

**실습 시작**

1. 실습을 시작하려면 페이지 상단에서 **실습 시작**을 선택합니다.

 계속 진행하기 전에 프로비저닝된 AWS 서비스가 준비될 때까지 기다려야 합니다.

1. 실습을 열려면 **콘솔 열기**를 선택합니다.

그러면 새 웹 브라우저 탭에서 자동으로 AWS Management Console에 로그인됩니다.

**별다른 지시가 없는 한 리전을 변경하지 마십시오.**

**일반적인 로그인 오류**

**오류: 우선 로그 아웃 필요**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**You must first log out before logging into a different AWS account**라는 메시지가 표시된다면 다음을 수행합니다.

* **click here**의 링크를 선택합니다.
* **Amazon Web Services Sign In** 웹 브라우저 탭을 닫고 초기 실습 페이지로 돌아갑니다.
* **콘솔 열기**를 다시 선택합니다.

**오류: 실습 시작 선택 시 아무 반응이 없음**

경우에 따라 일부 팝업 또는 스크립트 차단 웹 브라우저 확장 프로그램 때문에 **실습 시작** 버튼이 제대로 작동하지 않을 수 있습니다. 실습을 시작하는 데 문제가 있는 경우 다음을 수행합니다.

* 팝업 또는 스크립트 차단 프로그램의 허용 목록에 실습 도메인 이름을 추가하거나 차단 프로그램을 끕니다.
* 페이지를 새로 고친 후 다시 시도하십시오.

**과제 0: 개발 환경에 연결 및 IntelliJ 구성**

이 과제 세트에서는 개발 환경에 연결합니다. 연결되면 IntelliJ를 열고 이 실습에서 사용할 IDE를 구성합니다.

**과제 0.1: 개발 환경에 연결**

1. 아래 옵션 중에서 개발 환경에 연결할 방법을 선택합니다.

* [Apache Guacamole를 사용하여 Windows 개발 인스턴스에 연결](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connect_guacamole)
* [RDP를 사용하여 macOS 컴퓨터에서 Windows 개발 인스턴스에 연결](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connect_win_mac)
* [RDP를 사용하여 Windows 컴퓨터에서 Windows 개발 인스턴스에 연결](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connect_win_win)

**참고:** 후속 섹션의 실습 지침에서 명령을 실행해야 하는 경우 **IntelliJ 터미널**을 사용하십시오. 터미널의 사용을 기반으로 변수를 포함하여 특정 단계가 구성되었기 때문입니다.

**고려 사항:** 이 실습은 숙련된 개발자와 신입 개발자 모두를 위해 고안되었습니다.

* 도전을 즐기는 숙련된 개발자를 위해 각 과제에 앞서 과제 완료에 도움이 되는 충분한 정보가 담긴 **개략적 지침**이 제공됩니다.
* 업데이트를 완료하면 코드를 테스트하여 작동하는지 확인하고, 필요시 문제를 해결한 후 다음 과제를 진행하십시오.
* 신입 개발자는 실습의 각 단계를 자세히 설명하는 **상세 지침**을 참고할 수 있습니다.

**과제 0.2: IntelliJ IDE 구성**

1. 바탕 화면 아이콘을 사용하여 **IntelliJ**를 엽니다.

* **IntelliJ IDEA User Agreement** 창에서

I confirm that I have read and accept the terms of this User Agreement

 확인란을 선택하고 **Continue**를 선택합니다.

* **DATA SHARING** 창에서 **Don’t Send**를 선택합니다.

1. **Welcome** 화면에서  **Open** >

C:\code\java

 >

pom.xml

 > **OK** > **Open as Project** >  Trust projects in C:/code > **Trust Project**를 선택합니다.

**주의:** 기본 프로세스에서 플러그인/종속성 다운로드 및 기타 태스크를 완료할 때까지 기다리십시오. 완료를 기다리지 않을 경우 오류가 발생할 수 있습니다. 화면 하단에서 이 상태를 확인할 수 있습니다.

1. **Tip of the Day** 창을 닫습니다.
2. **AWS Explorer** 사용를 사용하기 위하여 화면 오른쪽 하단 또는 왼쪽 하단에서 **AWS Explorer** 탭을 열어서 Profile을

default

 로 사용하고

region

 을 지침 왼쪽의 값으로 설정합니다. ( 참고로 오른쪽 하단에 ‘AWS: default@us-west-2’ 와 같이 현재 설정된 정보가 보일것입니다. )

**참고:**

1. 이제 프로젝트에 SDK를 할당합니다. **File** > **Project Structure…** > **Project SDK:**

11 Amazon Corretto version 11.0.11

 > **OK**를 선택합니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **resources** >

config.properties

를 엽니다. 이 파일은 이 실습에서 각 프로그램의 변수를 저장하는 데 사용됩니다. 파일을 열어두십시오. 이후 과제에서 이를 참조하게 됩니다.

**과제 1: AWS CLI를 사용하여 Amazon DynamoDB 테이블 생성**

이 과제에서는 개발자 샌드박스와 **AWS CLI**를 사용하여 파티션 키, 정렬 키 및 프로비저닝된 처리량이 포함된 테이블을 생성합니다. 필요한 경우 Waiter를 사용하여 테이블의 가용성을 확보해야 합니다.

이 Amazon DynamoDB 테이블(참고)의 주요 액세스 패턴은 현재 사용자가 생성한 NoteId의 목록을 가져오는 것입니다. 따라서 **UserId**를 파티션 키로, **NoteId**를 정렬 키로 사용하는 테이블을 생성합니다.

1. **명령:** 하단에 있는 탭에서 **Terminal** 세션을 열고 다음 코드 조각을 실행하여 **notes** 테이블을 생성합니다.

aws dynamodb create-table ^

--table-name Notes ^

--attribute-definitions AttributeName=UserId,AttributeType=S AttributeName=NoteId,AttributeType=N ^

--key-schema AttributeName=UserId,KeyType=HASH AttributeName=NoteId,KeyType=RANGE ^

--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TableDescription:

AttributeDefinitions:

- AttributeName: NoteId

AttributeType: N

- AttributeName: UserId

AttributeType: S

CreationDateTime: '2022-04-07T16:09:50.658000+00:00'

ItemCount: 0

KeySchema:

- AttributeName: UserId

KeyType: HASH

- AttributeName: NoteId

KeyType: RANGE

ProvisionedThroughput:

NumberOfDecreasesToday: 0

ReadCapacityUnits: 5

WriteCapacityUnits: 5

TableArn: arn:aws:dynamodb:us-west-2:254879901662:table/Notes

TableId: 90eb6528-4b1b-41d2-a1a6-589d7866e79f

TableName: Notes

TableSizeBytes: 0

TableStatus: CREATING

**참고:** CreateTable은 비동기식 작업입니다. DynamoDB는 **CreateTable** 요청을 수신하는 즉시 **TableStatus**가 **CREATING**인 응답을 반환합니다. 테이블이 생성되면 DynamoDB가 **TableStatus**를 **ACTIVE**로 설정합니다. 읽기 및 쓰기 작업은 **ACTIVE** 테이블에서만 수행할 수 있습니다.

1. **명령:** 다음 명령을 실행하여 테이블이 생성될 때까지 기다리기 위해 **wait table-exists**를 사용합니다.

aws dynamodb wait table-exists --table-name Notes

**예상 출력:**

*오류가 발생하지 않는 한 없습니다.*

1. **명령:** 다음 명령을 실행하여 테이블의 현재 상태를 반환합니다.

aws dynamodb describe-table --table-name Notes | findstr TableStatus

**참고:** Waiter **table-exists**는 describe-table을 사용하여 **Table.TableStatus**가 **ACTIVE**를 반환할 때까지 20초마다 폴링합니다.

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TableStatus: ACTIVE

**참고:** 테이블의 상태가 **ACTIVE**로 업데이트될 때까지 출력이 *wait* 명령과 동일한 라인에서 일시 중지됩니다. 이후에는 다음 명령이 실행되고 테이블의 현재 상태가 출력됩니다.

 축하합니다! Amazon DynamoDB에서 테이블을 성공적으로 생성했습니다.

**과제 2: 테이블에 데이터 로드**

**Notes** 테이블을 생성했으므로 **notes.json** 데이터를 테이블로 가져오는 프로그램을 개발합니다. 이 파일에는 애플리케이션을 개발할 때 사용할 샘플 노트가 포함되어 있습니다. 파일은 이미 프로젝트 폴더에 저장되어 있습니다. **put-item** 작업을 사용하여 정의한 키 속성을 일치시킵니다.

**Pseudo code:** *notesLoadData.java*

* DynamoDB 서비스 클라이언트를 생성하고 입력 쿼리 파라미터를 읽습니다.
* 클라이언트에 대해 document API wrapper를 정의합니다.
* notes.json 데이터 파일을 반복적으로 읽고 UserId, NoteId 및 Note를 노드로 포함하여 항목을 빌드합니다.
* 키와 다른 속성 값을 매핑하여 putItem 작업을 실행합니다.
* 필요에 따라 오류와 예외를 확인합니다.

**개략적 지침**

* C:\code\java\notes.json

 파일에 제공된 초기 데이터를 검토합니다.

* 프로젝트에서

notesLoadData

 파일을 엽니다.

* 기본 구성을 포함하여 DynamoDB 서비스 클라이언트를 빌드합니다.
* 키 속성과 값을 사용하여 put-item에 대한 요청을 빌드합니다.
* 반복기를 사용하여 파일을 루프 처리하고 모든 레코드를 로드합니다.

**상세 지침**

**과제 2.1: put-item에 대한 요청 생성**

**putItem** 메서드는 입력 파일에서 각 노트를 **Notes** 테이블로 로드하는 데 사용됩니다. 노트가 이미 있는 경우에는 노트가 교체됩니다.

1. 먼저 **notes.json** 파일을 검토하여 테이블에 로드할 데이터를 확인합니다. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** >

notes.json

을 엽니다. 내용을 숙지한 후 이 파일을 닫습니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **java** > **dev.labs.dynamodb** >

notesLoadData

를 엽니다.

1. **복사/붙여넣기:** **TODO 0**을 찾고, 다음 코드를 추가하여 **DynamoDB client**를 빌드하고 **client**에 대한 **document API wrapper**를 정의합니다.

//Create DynamoDB client

AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()

.build();

//Use the DynamoDB document API wrapper

DynamoDB dynamoDB = new DynamoDB(client);

1. **복사/붙여넣기:** **TODO 1**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 **put-item**에 대한 요청을 빌드합니다.

* 보기 A

table.putItem(

new Item()

.withPrimaryKey("NoteId", noteId)

.withString("Note", note)

);

* 보기 B

table.putItem(

new Item()

.withPrimaryKey("UserId", userId, "NoteId", noteId)

.withString("Note", note)

);

1. 변경 사항을 저장합니다.

**추가 정보:** [put-item 메서드](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/JavaDocumentAPIItemCRUD.html#PutDocumentAPIJava)에 대해 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 1 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-1-solution) 섹션에서 찾을 수 있습니다.

**과제 2.2: 코드 테스트**

이제 코드를 실행하여 프로그램을 테스트하고 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesLoadData.java

 파일의 변경 사항을 저장해야 합니다.

1. **Build** > **Recompile 'notesLoadData.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. 다시 컴파일링된 후에는 이 파일을 닫을 수 있습니다.
2. **명령:** **터미널**에서 다음 명령을 사용하여 명령줄에서 **Maven**을 사용합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesLoadData"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Loading "Notes" table with data from file "notes.json"

PutItem succeeded: testuser 1 hello

PutItem succeeded: testuser 2 this is my first note

PutItem succeeded: newbie 1 Free swag code: 1234

PutItem succeeded: newbie 2 I love DynamoDB

PutItem succeeded: student 1 DynamoDB is NoSQL

PutItem succeeded: student 2 A DynamoDB table is schemaless

PutItem succeeded: student 3 PartiQL is a SQL compatible language for DynamoDB

PutItem succeeded: student 5 Maximum size of an item is \_\_\_\_ KB ?

PutItem succeeded: student 4 I love DynamoDB

 축하합니다! 테이블로 항목을 로드하는 데 성공했습니다.

**과제 3: 파티션 키 및 프로젝션을 사용하여 데이터 쿼리**

이 과제에서는 **Notes** 테이블에서 기존 데이터를 읽습니다. **Query** 작업을 사용하여 특정 사용자에 속하는 모든 노트를 가져오되 이름이 **NoteId** 및 **Note**인 2개 속성만 검색합니다.

테이블에서 데이터를 읽을 때 Amazon DynamoDB는 기본적으로 모든 항목 속성을 반환합니다. 모든 속성 대신 일부 속성만 가져오려면 **프로젝션 표현식**을 사용합니다. 단일 속성을 검색하려면 이름을 지정하십시오. 여러 속성인 경우 이름을 쉼표로 구분해야 합니다.

**Pseudo code:** *notesQuery.java*

* DynamoDB 서비스 클라이언트를 생성하고, Document API wrapper를 사용하고, 입력 쿼리 파라미터를 읽습니다.
* 테이블 정보를 가져옵니다.
* 빌드 요청: 기본 속성/값이 포함된 사양을 쿼리하여 원하는 항목을 일치시키되 **NoteId** 및 **Note**만 프로젝션합니다.
* 쿼리를 실행하고, 응답을 검색하고, 반복기를 사용하여 응답에서 각 항목을 처리합니다.

**개략적 지침**

* **KeyConditionExpression**을 사용하여 입력한 값과 파티션 키 속성(**UserId**)을 일치시키되 **NoteId** 및 **Note**만 프로젝션합니다.
* 쿼리를 실행하고 응답을 항목 컬렉션으로 검색합니다.
* 반복기를 사용하여 응답에서 각 항목을 처리합니다.

**상세 지침**

**과제 3.1: 파티션 키 및 프로젝션 표현식을 사용하여 QuerySpec 요청 빌드**

**UserId**를 기반으로 Notes 테이블을 쿼리하는 코드를 개발합니다. 이후 각 레코드에 대해 **NoteId** 및 **Note**를 프로젝션합니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **resources** >

config.properties

 파일을 엽니다.

1. **user\_id**에 설정된 값을 검토합니다. **student**로 설정되어 있어야 합니다. 이는 노트를 쿼리하는 데 사용하는 파티션 키입니다.
2. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **java** > **dev.labs.dynamodb** >

notesQuery

를 엽니다.

1. **복사/붙여넣기:** **TODO 2**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 파티션 키와 **withProjectExpression**을 사용하는 **QuerySpec** 요청을 빌드합니다. 사용자의 모든 노트를 읽고 **NoteId** 및 **Note**에 대한 속성만 검색해야 합니다.

* 보기 A

QuerySpec spec = new QuerySpec()

.withKeyConditionExpression("UserId = :v\_Id and NoteId = :v\_nId")

.withValueMap(new ValueMap()

.withString(":v\_Id", userId)

.withInt(":v\_nId", noteId));

* 보기 B

QuerySpec spec = new QuerySpec()

.withProjectionExpression("NoteId, Note")

.withKeyConditionExpression("UserId = :v\_Id")

.withValueMap(new ValueMap()

.withString(":v\_Id", userId));

**추가 정보:** [withKeyConditionExpression](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/QuerySpec.html" \t "_blank)을 사용하는 QuerySpec 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 2 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-2-solution) 섹션에서 찾을 수 있습니다.

**과제 3.2: 테이블 쿼리**

QuerySpec을 지정했으므로 응답을 검색합니다. Query 작업은 항상 결과 세트를 반환합니다. 일치하는 항목이 없는 경우 결과 세트는 비게 됩니다. 쿼리 결과는 항상 정렬 키 값으로 정렬됩니다. 쿼리 기준과 일치하는 항목의 어레이가 있습니다. 이 어레이의 각 요소는 속성 이름과 해당 속성의 값으로 구성되어 있습니다.

1. notesQuery

 파일을 계속 편집합니다.

1. **복사/붙여넣기:** **TODO 3**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 **QueryOutcome** 요청을 빌드합니다.

* 보기 A

ItemCollection<QueryOutcome> items = client.query(spec);

* 보기 B

ItemCollection<QueryOutcome> items = table.query(spec);

1. 변경 사항을 저장합니다.

**추가 정보:** [QuerySpec](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/QuerySpec.html" \t "_blank)에 대해 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 3 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-3-solution) 섹션에서 찾을 수 있습니다.

**과제 3.3: 코드 테스트**

이제 코드를 실행하여 프로그램을 테스트하고 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesQuery

 파일의 변경 사항을 저장해야 합니다.

1. **Build** > **Recompile 'notesQuery.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. 다시 컴파일링된 후에는 이 파일을 닫을 수 있습니다.
2. **명령:** **터미널**에서 다음 명령을 사용하여 명령줄에서 **Maven**을 사용합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesQuery"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query all notes belong to a user "student" and displaying "NoteId" and "Note" attributes only:

{

"NoteId" : 1,

"Note" : "DynamoDB is NoSQL"

}

{

"NoteId" : 2,

"Note" : "A DynamoDB table is schemaless"

}

{

"NoteId" : 3,

"Note" : "PartiQL is a SQL compatible language for DynamoDB"

}

{

"NoteId" : 4,

"Note" : "I love DynamoDB"

}

{

"NoteId" : 5,

"Note" : "Maximum size of an item is \_\_\_\_ KB ?"

}

**참고:** 결과에는 **NoteId** 및 **Note**만 포함되고 **UserId**는 포함되지 않습니다. 프로젝션 표현식에서 지정되지 않았기 때문입니다.

**축하합니다!** 이제 테이블에서 항목을 검색하는 방법을 알게 되었습니다.

**과제 4: Paginator를 사용하여 테이블 스캔**

이 과제에서는 **Notes** 테이블에서 모든 기존 데이터를 읽습니다. **Scan** 작업을 사용하여 전체 테이블을 읽고 노트의 일부인 특정 검색 텍스트를 사용하여 항목만을 필터링합니다.

**Scan** 작업은 테이블 또는 보조 인덱스의 **모든** 항목에 액세스하여 하나 이상의 항목 및 항목 속성을 반환합니다. DynamoDB가 반환하는 항목 수를 줄이려는 경우 **FilterExpression** 작업을 제공할 수 있습니다. DynamoDB는 **Scan** 작업의 결과를 *페이지로 배열*합니다. 페이지 배열을 사용하면 **Scan** 결과는 크기가 1MB(또는 미만)인 데이터 '페이지’들로 나뉩니다. 애플리케이션은 결과의 첫 번째 페이지를 처리한 다음, 두 번째 페이지를 처리하는 식으로 처리 작업을 계속 수행할 수 있습니다. 또한 단일 페이지에서 검색할 최대 항목 수를 제한할 수 있습니다.

**Pseudo code:** *notesScan.java*

* DynamoDB 클라이언트를 생성하고, Document API wrapper를 사용하고, Read 작업 입력을 초기화합니다.
* 노트의 검색 텍스트에 대해 필터 표현식을 사용하여 스캔 요청을 빌드합니다.
* 필터 표현식, 값, 프로젝션할 속성 목록을 사용하여 Scan 사양을 빌드합니다.
* 응답 페이지 크기를 제한합니다.
* 페이지 반복기를 사용하여 Scan 응답의 현재 페이지에서 각 항목을 처리합니다.

**개략적 지침**

* config.properties

 파일을 업데이트하고,

SQL

로 설정되도록 **searchText** 변수를 업데이트합니다.

* **Note**에 대한 withFilterExpression을 사용하여 Scan 사양을 빌드하고 **UserId**, **NoteId** 및 **Note**를 프로젝션합니다.
* 페이지 반복기를 사용하여 현재 페이지에서 각 항목을 처리합니다.

**상세 지침**

**과제 4.1: ScanSpec 빌드**

이 과제에서는 검색 기준을 **SQL**로 업데이트합니다. 그런 다음 updateSpec 요청을 빌드합니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **resources** >

config.properties

 파일을 엽니다.

1. SQL

로 설정되도록 **searchText** 변수를 업데이트하고, 이 파일에 변경 사항을 저장합니다. 각 노트가 이 값으로 검색됩니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **java** > **dev** > **labs** > **dynamodb** >

notesScan

 파일을 엽니다.

1. **복사:** **TODO 4**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 **withFilterExpression** 및 **withProjectionExpression**을 사용하는 **ScanSpec** 요청을 빌드합니다.

* 보기 A

ScanSpec scanSpec = new ScanSpec()

.withFilterExpression("contains (Note, :v\_txt)")

.withValueMap(new ValueMap().withString(":v\_txt", searchText))

.withProjectionExpression("UserId, NoteId, Note");

* 보기 B

ScanSpec scanSpec = new ScanSpec()

.withFilterExpression("begins\_with (Note, :v\_txt)")

.withValueMap(new ValueMap()

.withString(":v\_txt", searchText))

.withProjectionExpression("UserId, NoteId, Note");

**추가 정보:** 여기에서 [withFilterExpression](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/ScanSpec.html" \t "_blank)을 사용하는 ScanSpec 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 4 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-4-solution) 섹션에서 찾을 수 있습니다.

**과제 4.2: Paginator 구성**

이 과제에서는 pagination 값을 **1**로 설정합니다. 그러면 Scan 응답에서 페이지당 1개의 항목만 반환됩니다.

1. notesScan

 파일을 계속 편집합니다.

1. **TODO 5**의 코드를 업데이트하여 Paginator의 값을 **15**에서 새로운 값 **1**로 업데이트합니다.

**과제 4.3: Paginator 빌드**

이 과제에서는 Paginator를 빌드하여 현재 페이지의 각 항목을 처리합니다.

1. notesScan

 파일을 계속 편집합니다.

1. **복사/붙여넣기:** **TODO 6**의 코드를 다음 코드 조각으로 업데이트합니다.

Iterator<Item> item = page.iterator();

1. 변경 사항을 저장합니다.

**추가 정보:** [PageIterable](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/enhanced/dynamodb/model/PageIterable.html" \t "_blank)에 대해 자세히 알아보기

**과제 4.4: 코드 테스트**

이제 코드를 실행하여 스크립트를 테스트하고 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesScan

 파일의 변경 사항을 저장해야 합니다.

1. **Build** > **Recompile 'notesScan.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. 다시 컴파일링된 후에는 이 파일을 닫을 수 있습니다.
2. **명령:** **터미널**에서 다음 명령을 사용하여 명령줄에서 **Maven**을 사용합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesScan"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Scan table to list items with search text "SQL" as part of the note:

SCANNING TABLE...

Page: 1

Page: 2

Page: 3

{

"NoteId" : 1,

"UserId" : "student",

"Note" : "DynamoDB is NoSQL"

}

Page: 4

Page: 5

{

"NoteId" : 3,

"UserId" : "student",

"Note" : "PartiQL is a SQL compatible language for DynamoDB"

}

Page: 6

Page: 7

Page: 8

Page: 9

Page: 10

**참고:** **Limit** 파라미터를 사용하는 경우 단일 **Scan** 작업은 최대 수까지 항목 세트를 읽습니다. 최대 **1MB**의 데이터이며 **FilterExpression**을 사용하여 결과에 필터링을 적용합니다. 따라서 검색 텍스트 **SQL**이 일치하면 스캔된 응답에서 단 한 페이지가 반환되고 표시됩니다.

**참고:** **search\_text** 변수를 **DynamoDB**와 같은 다른 단어로 업데이트하여 어떤 레코드가 프로젝션되는지 확인할 수 있습니다.

 축하합니다! 이제 테이블에서 항목을 검색하는 방법을 알게 되었습니다.

**과제 5: 테이블의 항목 업데이트**

Table 객체의 updateItem 메서드를 사용하면 기존 속성 값을 업데이트하거나, 새 속성을 추가하거나, 기존 항목의 속성을 삭제할 수 있습니다. updateItem 메서드는 입력에서 지정한 항목 속성만을 수정하고, 항목의 다른 속성은 변경되지 않습니다. 업데이트 전후에 표시되는 항목 속성에 액세스하려면 *ReturnValues*를 사용하십시오.

이 과제에서는 특정 사용자 노트에 새 속성 **Is\_Incomplete**를 추가하고 값을 **Yes**로 지정합니다. 나중에 **Is\_Incomplete** = **Yes**라는 조건을 추가하여 동일한 항목을 다른 노트 텍스트로 업데이트합니다. 업데이트 작업 후에 항목 속성을 반환하게 됩니다.

**Pseudo code:** *notesUpdate.java*

* DynamoDB 서비스 클라이언트를 생성하고 입력 쿼리 파라미터를 읽습니다.
* *withUpdateExpression*을 사용하여 update spec 요청을 빌드하고 실행하여 키를 사용하고 표현식을 업데이트합니다. 새 속성을 추가하고 값을 할당합니다.
* *withConditionExpression*을 사용하여 update spec 요청을 빌드하고 실행합니다. 조건과 일치하는 경우에만 새 노트 텍스트로 항목을 업데이트합니다.
* 업데이트 조건이 일치하지 않는 경우 *ConditionalCheckFailedException* 표현식을 캡처합니다.
* 필요에 따라 결과 및 오류를 표시합니다.

**개략적 지침**

* update specification을 빌드하여 **Is\_Incomplete** 속성을 추가하고 일치하는 키에 값 **Yes**를 할당합니다. 업데이트 작업 후에 모든 속성을 반환합니다.
* update specification을 빌드하여 **Is\_Incomplete**에 **Yes**가 할당된 경우에만 일치하는 키에 대해 **Note** 및 **Is\_Incomplete** 속성을 업데이트합니다. 업데이트 작업 후에 업데이트된 속성만 반환합니다.
* incomplete인 경우에만 노트 텍스트 업데이트를 허용합니다.
* 실패한 조건부 확인에 대해 업데이트 작업에서 예외를 캡처합니다.

**상세 지침**

**과제 5.1: 새 속성 추가**

이 과제에서는 특정 사용자 노트에 새 속성 **Is\_Incomplete**를 추가하고 값에 **Yes**를 할당합니다. 사용자 **student**에 속하고 노트 ID가 **5**인 항목을 사용합니다. 구성 파일에서 항목 키를 업데이트한 다음 요청을 빌드하여 이 항목을 업데이트합니다. 업데이트 작업 후에 모든 속성을 반환합니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **resources** >

config.properties

 파일을 엽니다.

1. config.properties

 파일을 **note\_id** 변수 **5**로 업데이트하고 변경 사항을 저장합니다.

1. **Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **java** > **dev** > **labs** > **dynamodb** >

notesUpdate

 파일을 엽니다.

1. **복사/붙여넣기:** **TODO 7**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 **withUpdateExpression**을 사용하는 **UpdateSpec** 요청을 빌드합니다.

* 보기 A

UpdateItemSpec updateItemSpec = new UpdateItemSpec()

.withPrimaryKey("UserId", userId, "NoteId", noteId)

.withUpdateExpression("set #inc = :val1")

.withNameMap(new NameMap()

.with("#inc", "Is\_Incomplete"))

.withValueMap(new ValueMap()

.withString(":val1", "Yes"))

.withReturnValues(ReturnValue.ALL\_NEW);

* 보기 B

UpdateItemSpec updateItemSpec = new UpdateItemSpec()

.withPrimaryKey("UserId", userId, "NoteId", noteId)

.withUpdateExpression("set #inc = :val1")

.withNameMap(new NameMap()

.with("#inc", "Is\_Incomplete"))

.withValueMap(new ValueMap()

.withString(":val1", "Yes"))

.withReturnValues(ReturnValue.ALL\_OLD);

**추가 정보:** [withUpdateExpression](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/UpdateItemSpec.html" \t "_blank)을 사용하는 UpdateSpec 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 7 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-7-solution) 섹션에서 찾을 수 있습니다.

**과제 5.2: 코드 테스트**

이제 코드를 실행하여 스크립트를 테스트하고 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesUpdate

 파일의 변경 사항을 저장해야 합니다.

1. notesUpdate

 파일에 변경 사항을 저장합니다.

1. **Build** > **Recompile 'notesUpdate.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. **명령:** **터미널**에서 다음 명령을 사용하여 명령줄에서 **Maven**을 사용합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesUpdate"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

UPDATE#1: Printing item after adding the new attribute "Is\_Incomplete" :

{

"NoteId" : 5,

"Is\_Incomplete" : "Yes",

"UserId" : "student",

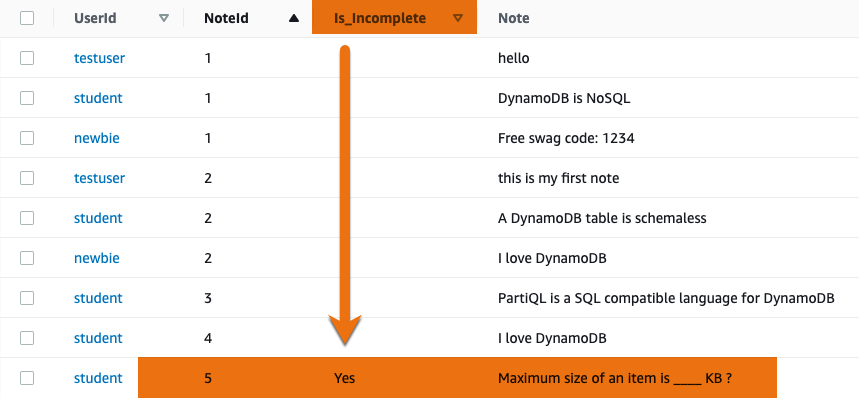
"Note" : "Maximum size of an item is \_\_\_\_ KB ?"

}

업데이트 작업 후 응답에 새 속성 **Is\_Incomplete**가 **Yes**로 추가되었고 모든 속성이 반환되었습니다. 이 항목의 노트 텍스트는 공백 때문에 불완전해 보입니다. 다음 과제에서 완전한 텍스트로 업데이트합니다.

1. **DynamoDB** 콘솔에서 이를 확인할 수 있습니다.

* **DynamoDB** 콘솔 > **Tables** > Notes > **Actions** > *Explore Items* > **View Items** >  **Notes** **Scan** > **Notes** > **Run**을 엽니다.



 축하합니다! 이제 테이블에서 특정 속성을 업데이트할 수 있습니다.

**과제 5.3: 조건부로 항목 업데이트**

이 과제에서는 동일한 항목을 새 노트 텍스트로 업데이트하고 **Is\_Incomplete**을 **Is\_Incomplete**이 **Yes**로 설정된 경우에만 **No**로 설정합니다. 업데이트 작업 후에 업데이트된 항목 속성만 반환합니다.

1. **new\_note** 변수 값이 다음 값과 같도록

config.properties

 파일을 편집합니다.

Maximum item size in DynamoDB is 400 KB

1. 변경 사항을

config.properties

 파일에 저장합니다.

1. notesUpdate

 파일을 계속 편집합니다.

1. **복사/붙여넣기:** **TODO 8**의 코드를 다음 코드 조각으로 업데이트하여 **Is\_Complete** 속성이 평가되도록 합니다.

.withConditionExpression("Is\_Incomplete = :v\_old")

**추가 정보:** [withConditionExpression](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/UpdateItemSpec.html" \t "_blank)을 사용하는 UpdateSpec 자세히 알아보기

**과제 5.4: 조건부 확인 실패 예외 캡처하기**

이전 단계에서는 업데이트가 발생하기 위해 충족되어야 하는 조건을 포함하여 updateItem 메서드에 대한 선택적 파라미터를 지정했습니다. 지정한 조건이 충족되지 않는 경우 Java용 AWS SDK에서 조건부 확인 실패 예외가 발생합니다. 이 과제에서는 이 실패 예외를 캡처합니다.

1. notesUpdate

 파일을 계속 편집합니다.

1. **복사/붙여넣기:** **TODO 9**의 코드를 아래에 있는 올바른 코드 조각으로 업데이트하여 **ConditionalCheckFailedException**에 대한 **catch**를 빌드합니다.

* 보기 A

catch (ConditionalCheckFailedException e) {

System.out.println("\nUPDATE#2 - REPEAT: Printing item after the conditional update for the item - \"" + userId + "\" and \"" + noteId + "\" - FAILURE:");

System.out.println("UpdateItem failed on item due to unmatching condition!");

System.err.println(e.getMessage());

}

* 보기 B

catch (TransactionConflictException e) {

System.out.println("\nUPDATE#2 - REPEAT: Printing item after the conditional update for the item - \"" + userId + "\" and \"" + noteId + "\" - FAILURE:");

System.out.println("UpdateItem failed on item due to unmatching condition!");

System.err.println(e.getMessage());

}

**추가 정보:** [ConditionalCheckFailedException](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/model/ConditionalCheckFailedException.html" \t "_blank)을 사용하는 UpdateSpec 자세히 알아보기

**해답:** 이 단계의 해답은 이 지침 하단의 [TODO 9 해답](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-9-solution) 섹션에서 찾을 수 있습니다.

**과제 5.5: 코드 테스트**

이 과제에서는 프로그램을 실행하여 조건부 업데이트가 예상대로 작동하는지 확인합니다. *updateExistingAttributeConditionally* 메서드를 두 번 호출하여 조건부 확인 실패 효과를 확인합니다. 첫 번째 호출은 항목을 새 텍스트로 업데이트하고, 두 번째 호출은 오류와 함께 업데이트가 실패합니다.

1. 다음으로 아래에 있는 코드를 추가하여 조건부 확인을 사용하여 업데이트 작업을 호출합니다.
2. **복사/붙여넣기:** **TODO 10**에 대한 기존 코드를 다음 코드 조각으로 업데이트합니다.

//Allow update to the Notes item only if the note is incomplete - SUCCESS

updateExistingAttributeConditionally(table, qUserId, qNoteId, newNote);

//Allow update to the Notes item only if the note is incomplete - FAILURE

updateExistingAttributeConditionally(table, qUserId, qNoteId, newNote);

1. notesUpdate

 파일에 변경 사항을 저장합니다.

이제 프로그램을 실행하여 프로그램이 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesUpdate

 파일의 변경 사항을 저장해야 합니다.

1. **Build** > **Recompile 'notesUpdate.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. **명령:** **Maven**을 사용하여 코드를 다시 실행하고 이 값이 이미 업데이트되었을 때 어떻게 응답하는지 확인합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesUpdate"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

UPDATE#1: Printing item after adding the new attribute "Is\_Incomplete" :

{

"NoteId" : 5,

"Is\_Incomplete" : "Yes",

"UserId" : "student",

"Note" : "Maximum size of an item is \_\_\_\_ KB ?"

}

UPDATE#2: Printing item after the conditional update for the item - "student" and "5" - SUCCESS:

{

"Is\_Incomplete" : "No",

"Note" : "Maximum item size in DynamoDB is 400 KB"

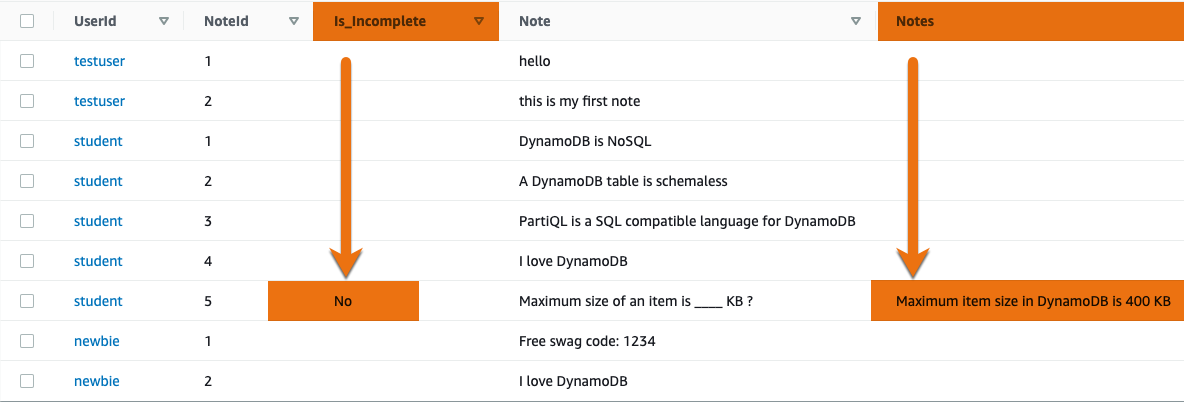
}

UPDATE#2 - REPEAT: Printing item after the conditional update for the item - "student" and "5" - FAILURE:

UpdateItem failed on item due to unmatching condition!

The conditional request failed (Service: AmazonDynamoDBv2; Status Code: 400; Error Code: ConditionalCheckFailedException; Request ID: IEKQI6JDQJPV5T4451GDA0E49NVV4KQNSO5AEMVJF66Q9ASUAAJG; Proxy: null)

첫 번째 업데이트 호출은 조건이 일치하여 성공했으며 업데이트된 속성을 예상대로 반환했습니다. 두 번째 업데이트 호출은 조건이 일치하지 않아 실패했습니다.



 축하합니다! 이제 테이블에서 특정 속성을 업데이트할 수 있습니다. 또한 조건부로 항목을 업데이트할 수 있음을 확인했습니다.

**선택 도전 과제: CRUD 작업에 대해 DynamoDBMapper 클래스 사용**

이 과제에서는 개체 지속성 모델을 사용하여 DynamoDB 서비스와 상호 작용하는 새 프로그램을 개발하는 도전 과제가 주어집니다.

Java용 AWS SDK는 DynamoDBMapper 클래스를 제공하여 클라이언트 측 클래스를 Amazon DynamoDB 테이블에 매핑할 수 있습니다. DynamoDBMapper를 사용하려면 DynamoDB 테이블의 항목과 코드의 해당하는 개체 인스턴스 사이의 관계를 정의합니다. DynamoDBMapper 클래스를 사용하면 테이블에 액세스하고, 다양한 CRUD(생성, 읽기, 업데이트 및 삭제) 작업을 수행하고, 쿼리를 실행할 수 있습니다. DynamoDBMapper 클래스를 사용하여 테이블에 액세스하고, 다양한 CRUD 작업을 수행하고, 쿼리를 실행합니다.

[DynamoDB Mapper](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBMapper.html) 클래스 및 메서드를 사용하여 테이블 구조 및 스키마를 정의합니다.

**개략적 지침**

* 기본 구성을 포함하여 DynamoDB 서비스 클라이언트를 정의합니다.
* DynamoDBMapper 인터페이스를 DynamoDB 클라이언트에 대한 래퍼로 빌드합니다.
* 주석을 포함한 NotesItems 클래스를 정의하여 개체 필드를 DynamoDB 속성 이름(UserId, NoteId 및 Note)에 매핑합니다.
* NotesItems 클래스를 인스턴스화하고 DynamoDBMapper 메서드를 사용하여 CRUD 작업을 수행합니다.
* load(), save() 및 delete() 메서드를 사용하여 CRUD 작업을 수행합니다.

**상세 지침**

**과제 6.1: 스크립트 개발**

**Project** 창에서 **java [Lab-DynamoDB] C:\code\java** > **src** > **main** > **java** > **dev** > **labs** > **dynamodb** > 이름이

notesCRUDmapper.java

인 스켈레톤 파일을 엽니다. 시작에 도움이 되도록 파일이 이미 생성되어 있습니다. 이 파일을 엽니다.

**추가 정보:** 지원이 필요한 경우 **Solutions** 폴더의

notesCRUDmapper-solution.java

 파일에 있는 **Solutions** 폴더에서 전체 솔루션을 찾을 수 있습니다.

1. 코드가 만족스럽다면 다음 단계는 이를 테스트하는 것입니다.

**과제 6.2: 코드 테스트**

이제 코드를 실행하여 스크립트를 테스트하고 예상대로 작동하는지 확인합니다. 구문 오류가 없어야 하고

notesCRUDmapper.java

 파일의 변경 사항을 저장해야 합니다.

1. **Build** > **Recompile 'notesCRUDmapper.java**’를 선택합니다.

**참고:** 하단에 있는 상태 표시줄을 검토하고 **Recompiling**이 완료되는지 확인한 다음 진행하십시오.

1. **명령:** **터미널**에서 다음 명령을 사용하여 명령줄에서 **Maven**을 사용합니다.

mvn -q exec:java -Dexec.mainClass="dev.labs.dynamodb.notesCRUDmapper"

**예상 출력:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Item retrieved:

Notes [User=testuser, Note Id=1, Notes=this is my very first note]

Item updated:

Notes [User=testuser, Note Id=1, Notes=updated notes]

Retrieved the previously updated item:

Notes [User=testuser, Note Id=1, Notes=updated notes]

deleting the previously existing item:

Notes [User=testuser, Note Id=1, Notes=updated notes]

Done - Sample item is deleted.

CRUD operations complete!

Item retrieved:

Notes [User=newbie, Note Id=1, Notes=Free swag code: 1234]

Item updated with new notes:

Notes [User=newbie, Note Id=1, Notes=free swag registration code 380765988]

**요약**

**축하합니다! 이 실습을 완료했습니다.** 이제 **Java** 버전에서 다음을 수행할 수 있습니다.

* 프로그램의 하위 수준, 문서 및 상위 수준 API를 사용하여 프로그래밍 방식으로 DynamoDB와 상호 작용합니다.
* 파티션 키, 정렬 키 및 적합한 프로비저닝된 처리량을 포함한 Waiter를 사용하여 테이블을 생성합니다.
* 파일에서 JSON 객체를 읽고 테이블을 로드합니다.
* 키 속성, 필터, 표현식 및 페이지 배열을 기반으로 테이블에서 항목을 검색합니다.
* 새 속성을 추가하고 조건부로 데이터를 변경하여 항목을 업데이트합니다.
* 해당되는 경우 PartiQL 및 객체 지속성 모델을 사용하여 DynamoDB 데이터에 액세스합니다.

**실습 종료**

다음 단계에 따라 콘솔을 닫고 실습을 종료합니다.

1. **AWS Management Console**로 돌아갑니다.
2. 페이지 오른쪽 상단에서 **AWSLabsUser**를 선택하고 **Sign out**을 선택합니다.
3. **실습 종료**를 선택한 다음 실습을 종료할 것임을 확인합니다.

AWS Training and Certification에 대한 자세한 내용은 [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/)을 참조하십시오.

*여러분의 피드백을 환영합니다.* 피드백, 제안 사항 또는 수정 요청 사항을 제공하려면 [*AWS Training and Certification 문의 양식*](https://support.aws.amazon.com/#/contacts/aws-training)에 세부 정보를 입력해 주시기 바랍니다.

**추가 자료**

* [put-item 메서드에 대해 자세히 알아보기](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/JavaDocumentAPIItemCRUD.html#PutDocumentAPIJava)
* [withKeyConditionExpression을 사용하는 QuerySpec 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/QuerySpec.html)
* [QuerySpec에 대해 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/QuerySpec.html)
* [withFilterExpression을 사용하는 ScanSpec 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/ScanSpec.html)
* [PageIterable에 대해 자세히 알아보기](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/enhanced/dynamodb/model/PageIterable.html)
* [withUpdateExpression을 사용하는 UpdateSpec 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/UpdateItemSpec.html)
* [withConditionExpression을 사용하는 UpdateSpec 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/document/spec/UpdateItemSpec.html)
* [ConditionalCheckFailedException을 사용하는 UpdateSpec 자세히 알아보기](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/dynamodbv2/model/ConditionalCheckFailedException.html)
* [DynamoDB에 대해 자세히 알아보기](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBMapper.html)

**코드 도전 과제 해답**

**TODO 1 해답**

* 복합 키가 불완전하므로 보기 A는 올바르지 않습니다. **Notes** 테이블은 파티션 및 정렬 키를 모두 사용하여 항목을 식별합니다. 파티션 키 및 정렬 키가 있는 테이블에서 두 항목이 동일한 파티션 키 값을 갖는 것이 가능합니다. 하지만 두 항목의 정렬 키 값은 달라야 합니다.
* **보기 B가 올바른 코드 조각입니다.** 항목을 테이블에 추가할 때 파티션 키 속성이 필요합니다.

table.putItem(

new Item()

.withPrimaryKey("UserId", userId, "NoteId", noteId)

.withString("Note", note)

);

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-1)

**TODO 2 해답**

* 보기 A는 올바르지 않습니다. 이 옵션에 사용된 QuerySpec은 파티션 키와 정렬 키 값 모두로 단 하나의 항목만 반환합니다.
* **보기 B가 올바른 코드 조각입니다.** 쿼리는 해당 파티션 키 값을 가진 모든 항목을 반환합니다. 선택적으로 정렬 키 속성을 제공하고 비교 연산자를 사용하여 보기 A와 유사하게 검색 결과를 세분화할 수 있습니다. 이 경우 noteID를 전달함으로써 사용자 student에 속하는 다른 노트를 제외합니다.

QuerySpec spec = new QuerySpec()

.withProjectionExpression("NoteId, Note")

.withKeyConditionExpression("UserId = :v\_Id")

.withValueMap(new ValueMap()

.withString(":v\_Id", userId));

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-2)

**TODO 3 해답**

* DynamoDB 클라이언트를 사용하는 쿼리에는 하위 수준 요청이 필요하고 QueryResult 개체로서 하위 수준 응답을 반환하므로 보기 A가 올바르지 않습니다.
* **보기 B가 올바른 코드 조각입니다.**

ItemCollection<QueryOutcome> items = table.query(spec);

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-3)

**TODO 4 해답**

* **보기 A가 올바른 코드 조각입니다.**

ScanSpec scanSpec = new ScanSpec()

.withFilterExpression("contains (Note, :v\_txt)")

.withValueMap(new ValueMap().withString(":v\_txt", searchText))

.withProjectionExpression("UserId, NoteId, Note");

* 올바르지 않은 필터 표현식 함수 **begins with**를 사용하므로 보기 B는 올바르지 않습니다. 이 함수는 속성 값의 첫 문자 몇 개를 확인합니다.

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-4)

**TODO 7 해답**

* **보기 A가 올바른 코드 조각입니다.**

UpdateItemSpec updateItemSpec = new UpdateItemSpec()

.withPrimaryKey("UserId", userId, "NoteId", noteId)

.withUpdateExpression("set #inc = :val1")

.withNameMap(new NameMap()

.with("#inc", "Is\_Incomplete"))

.withValueMap(new ValueMap()

.withString(":val1", "Yes"))

.withReturnValues(ReturnValue.ALL\_NEW);

* 업데이트 작업 이후 항목의 ‘오래된’ 값만 반환하므로 보기 B는 올바르지 않습니다.

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-7)

**TODO 9 해답**

* **보기 A가 올바른 코드 조각입니다.**

catch (ConditionalCheckFailedException e) {

System.out.println("\nUPDATE#2 - REPEAT: Printing item after the conditional update for the item - \"" + userId + "\" and \"" + noteId + "\" - FAILURE:");

System.out.println("UpdateItem failed on item due to unmatching condition!");

System.err.println(e.getMessage());

}

* 동일한 항목에 대해 진행 중인 트랜잭션이 있을 때 TransactionConflictException 예외가 발생하므로 보기 B는 올바르지 않습니다.

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#todo-9)

**RDP 연결 옵션**

**Apache Guacamole를 사용하여 Windows 개발 인스턴스에 연결**

1. **Connection Details** 섹션에서 **GuacamoleLink**를 복사하여 브라우저에 붙여넣습니다.
2. 브라우저에서 Apache Guacamole 로그인 페이지로 이동하고 다음 단계에 따라 로그인합니다.

* **Username**에

student

를 입력합니다.

* **Password**에 이 지침의 왼쪽에 있는 **AdministratorPassword** 값을 입력합니다.
* **Log In**을 선택합니다.

원격 인스턴스에 대한 연결이 곧 시작됩니다. 연결을 열면 개발 인스턴스 데스크톱의 이미지가 표시됩니다. 일반 데스크톱 또는 원격 데스크톱 클라이언트에서처럼 이 이미지와 상호 작용할 수 있습니다.

이제 브라우저에서 Guacamole을 사용해 Windows 개발 인스턴스에 연결되었습니다.

**참고:** 네트워크 팝업 **Do you want to allow your PC to be discoverable by other PCs and devices on this network**?가 표시됩니다. **No**를 선택합니다.

1. **참고:** ***Guacamole 브라우저 세션을 사용할 때 복사하고 붙여넣는 방법***

* Windows에서 **Ctrl -> Alt -> Shift**를 선택하거나 macOS에서 **command** -> **control** -> **shift**를 선택하여 클립보드 편집기를 엽니다.
* 실습 지침에 있는 텍스트를 복사하고 세션의 클립보드 편집기에 붙여넣습니다.
* Windows에서 **Ctrl -> Alt -> Shift**를 선택하거나 macOS에서 **command** -> **control** -> **shift**를 선택하여 클립보드 편집기를 닫습니다.
* 이제 클립보드에서 세션의 일반 붙여넣기 명령을 사용할 수 있습니다. 또한 클립보드를 편집하거나 이후 복사본으로 바꿀 수도 있습니다.

1. [지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connection-options)

**Windows 컴퓨터에서 Windows 개발 인스턴스에 연결**

1. 이 지침 왼쪽의 탐색 창에 있는 **WindowsInstanceIP**를 클립보드에 복사합니다.
2. 컴퓨터에서 원격 데스크톱 연결 애플리케이션을 엽니다.

* **시작** 아이콘을 선택하고 **검색** 아이콘을 선택합니다.

Remote Desktop Connection

을 입력합니다. **Programs** 목록에 해당 애플리케이션이 표시되면 선택합니다.

1. **Computer**에 Windows 인스턴스의 **IP**를 붙여넣습니다.
2. **Connect**를 선택합니다.
3. 원격 데스크톱 연결에 사용자 이름과 암호를 요구하는 로그인 대화 상자가 표시됩니다. 기본적으로 원격 데스크톱 연결 애플리케이션은 현재 Windows 사용자 이름과 도메인을 사용합니다. 이를 변경하려면 **Use another account**를 선택합니다.

**참고:** Windows 10에서는 **More Choices**를 선택한 후에 **Use a different account**를 선택합니다.

1. 이 지침의 왼쪽에 있는 탐색 창에서 **AdministratorPassword**를 복사합니다.
2. 로그인 보안 인증에 다음 값을 사용합니다.

* **User name**에

\Administrator

를 입력합니다.

* **Password**에 클립보드의 암호를 붙여넣습니다.

**참고:** 사용자 이름에 포함된

\

 기호는 원격 데스크톱 연결에 도메인 사용자가 아닌 로컬 관리자로서 로그인하고 있다는 뜻이므로 중요합니다.

1. 인스턴스에 연결하려면 **OK**를 선택합니다. 연결 확인에 사용된 인증서가 신뢰할 수 있는 알려진 루트 인증서가 아니라는 메시지가 표시되면 **Yes**를 선택합니다.

**결과**

원격 인스턴스에 대한 연결이 곧 시작됩니다.

**참고:** 네트워크 팝업 **Do you want to allow your PC to be discoverable by other PCs and devices on this network**?가 표시됩니다. **No**를 선택합니다.

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connection-options)

**macOS 컴퓨터에서 Windows 개발 인스턴스에 연결**

이 섹션에서는 macOS 컴퓨터에서 Windows Amazon EC2 인스턴스에 연결합니다.

1. Microsoft 원격 데스크톱을 설치합니다(아직 설치되지 않은 경우). 설치하려면 다음 작업을 완료하십시오.

* 화면 상단에서 Apple 아이콘을 선택합니다.
* **About This Mac**을 선택합니다.
* macOS 버전을 확인합니다.
* Dock에서 **App store**를 기동합니다.
* 다음 문자열을 검색합니다.

Microsoft Remote Desktop

* **Install** 또는 **GET**을 선택하여 해당 버전을 설치합니다.
* 이 글을 쓰는 시점에서 현재 macOS 버전은 OSX 10.15.7이지만 실습에서는 Microsoft Remote Desktop 버전 10.5.2를 설치합니다. 연결 지침은 이 버전을 기반으로 합니다.

1. **Microsoft Remote Desktop**을 열려면 Dock에서 **Launchpad**를 선택합니다. 그런 다음 **Microsoft Remote Desktop**을 선택합니다.
2. 새 연결을 만들려면 상단 메뉴에서 **Connections** > **Add PC**를 선택하고 다음 필드를 채웁니다.

* **PC name:** *이 지침의 왼쪽에 있는 Lab 속성에서 복사한 WindowsInstanceIP를 붙여넣습니다.*
* **User account:**

Administrator

* **Friendly name:**

Windows Dev Instance

* **Reconnect if the connection is dropped:** *선택되지 않은 경우 이 옵션을 선택합니다.*
* **Add**를 선택합니다.

1. 이 지침의 왼쪽에 있는 탐색 창에서 **WindowsPassword**를 클립보드에 복사합니다.
2. Microsoft Remote Desktop 연결 창으로 돌아가 **Windows 개발 인스턴스** 연결에 대한 컨텍스트 메뉴를 열고 **Open**을 선택합니다.
3. 다음 메시지가 표시됩니다. **You are connecting to the RDP host “IP.Address.Listed.Here”. There certificate couldn’t be verified back to a root certificate. Your connection may not be secure. Do you want to continue?** **Continue**를 선택합니다.
4. 암호를 묻는 메시지가 나타나면 클립보드에 복사한 암호를 입력하고 **Continue**를 선택합니다.

**결과**

원격 인스턴스에 대한 연결이 곧 시작됩니다.

**참고:** 네트워크 팝업 **Do you want to allow your PC to be discoverable by other PCs and devices on this network**?가 표시됩니다. **No**를 선택합니다.

[지침으로 돌아가기](https://classrooms.labs.aws.training/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A006961644361%3Ablueprintversion%2FILT-TF-200-DODEVA-4%2Flab-3-java%3A4.5.10-07cc3a97/ko-KR/e8919513-39ae-48d9-af22-3a606f897383::ajQFLkuDdRyM1NTR7cadZK#connection-options)

**부록**