Dynamo DB와 RDB의 차이

작성자 : 김영훈

작성일 : 2023.06.20

버전 : 0.1

# 1. Dynamo DB란?

Dynamo DB는 키-값 및 도큐먼트 자료구조를 지원하는 NoSQL 데이터베이스이다



# 2. SQL과 NoSQL에 차이점

SQL은 관계형 데이터베이스(RDB)를 제어하는데 사용하는 언어이다.

RDB의 경우 Schema가 존재하여 Schema 단위로 내부에 테이블을 만들어서 정해진 테이블의 규격에 맞춰서 Insert, Update, Select등의 쿼리를 수행할 수 있는 방면,

NoSQL은 Not Only SQL이라는 뜻으로 SQL처럼 정해진 테이블 형식을 따르지 않는다는 차이점이 있다.

# 3.NoSQL을 사용하는 이유

1. 수평적 확장에 유리하다.
2. 속도가 빠르다
3. 유연하다

# 4. NoSQL의 단점

1. Join 연산이 불가능하다.
2. Attribute에 대해서 상태검사를 수행할 때는 SQL보다 느릴 수 있다.
3. 두번째에 단점 때문에 설계 이전에 테이블을 어떻게 사용할지에 대해서 미리 고민을 해야한다.

# 5. Dynamo DB JavaScript SDK와 RDB 코드 비교

- 가상의 Table인 userlist 테이블, 테이블에 column은 userid(pk), name, age, job이 존재한다.

|  |  |
| --- | --- |
| RDB | Dynamo DB |
| SELECT | Scan |
| INSERT | Put |
| UPDATE | Update( UpdateExpression)  (전체 변경x, key가 있어야 업데이트 가능) |
| DELETE | Delete  (테이블 삭제나 key가 있거나 또는 sort키를 설정하여 여러행 삭제 가능) |
| WHERE | FilterExpression |
| GROUP BY | GROUP BY기능 지원 안함 |
| ORDER BY | Sort key를 설정해야 사용 가능 |
| DISTINCT | DISTINCT기능 지원 안함 |
| JOIN | JOIN기능 지원 안함 |
| LIKE | Contains(대소문자 구분필요) |

## SELECT

1. userlist에 모든 데이터 검색 쿼리.

* **SQL**

SELECT \* FROM userlist;

* **Dynamo DB**

dynamo.scan({

TableName: ‘userlist’

});

1. Userlist에 name, age 데이터 검색 쿼리

* **SQL**

SELECT name, age FROM userlist;

* **Dynamo DB**

dynamo.scan({

TableName: ‘userlist’,

ProjectionExpression: “name, age”

});

1. userlist에서 userid(pk)가 100003인 데이터 검색 쿼리

* **SQL**

SELECT \* FROM userlist WHERE userid = 100003;

* **Dynamo DB**

dynamo.scan({

TableName: userlist,

FilterExpression: “userid = :useridVal””,

ExpressionAttributeValues: {“: useridVal” : 100003}

});

1. userlist에서 age가 13살 이상인 데이터 검색 쿼리

* **SQL**

SELECT \* FROM userlist WHERE age >= 13;

* **Dynamo DB**

dynamo.scan({

TableName: ‘userlist’,

FilterExpression: “age >= :ageVal”,

ExpressionAttributeValues: {“:ageVal” : 13}

});

1. userlist에서 이름이 지민이고 age가 20살 이상인 데이터 검색 쿼리

* **SQL**

SELECT \* FROM userlist WHERE age >= 20 NAME LIKE ‘%지민’;

* **Dynamo DB**

dynamo.scan({

TableName: ‘userlist’,

FilterExpression: “age >= :ageVal AND contains(name, ‘지민’)”,

ExpressionAttributeValues: {“:ageVal” : 13}

});

## INSERT

1. INSERT문

* **SQL**

INSERT INTO userlist (name, age, job ) VALUES (홍지수, 24, 간호사);

* Dynamo DB(ConditionEpression은 데이터가 중복된 고유값이 있으면 생성하지 않음)

dynamo.put({

TableName: ‘userlist’,

ConditionExpression: "attribute\_not\_exists(userid)”,

Item: {

userid: ‘100001’,

name: ‘홍지수’,

age: 24,

job: ‘간호사’

}

});

## UPDATE

1. Userid가 10003인 데이터에 직업을 의사로 변경

* **SQL**

UPDATE userlist SET job = ‘의사’ WHERE userid = 10003;

* Dynamo DB

dynamo.update({

TableName: ‘userlist’,

Key: {

userid: “10003”;

},

UpdateExpression: “set job = :jobVal“,

ExpressionAttributeValues: {

“:jobVal”: “의사”

}

});

## DELETE

1. userid 10003 데이터 삭제

* **SQL**

DELETE FROM userlist WHERE userid = 10003;

* **Dynamo DB**

dynamo.delete({

TableName: ‘userlist’,

Key: {

userid: 10003

}

});