Описание выбранного решения

Amazon DynamoDB — это управляемая NoSQL-база данных от AWS, предназначенная для высокопроизводительных, масштабируемых и отказоустойчивых решений. Она была разработана как преемник Amazon SimpleDB и представлена в 2012 году. Основной причиной её создания стало стремление Amazon к решению проблем масштабируемости и производительности, с которыми сталкивались реляционные базы данных в условиях растущей нагрузки и глобального распределения данных.

Основные особенности выбранного решения

Тип NoSQL в контексте DynamoDB

DynamoDB относится к категории **ключ-значение (Key-Value) и документоориентированных (Document)** NoSQL-баз данных. Это значит, что данные хранятся как наборы ключей и значений и поиск по ключу работает за O(1), но при этом значением может быть сложная структура

Этот подход дает **гибкость в моделировании данных**:

- Можно работать как с простыми ключ-значение объектами (аналогично Redis).
- Можно хранить сложные структуры (аналогично MongoDB).

Принципы работы и механизмы DynamoDB

1. DynamoDB — полностью управляемая БД.

- **Пользователю не нужно** настраивать, обновлять, патчить или масштабировать серверы.
- Можно **развернуть локально** для тестирования, используя DynamoDB Local (Docker).

2. Автоматическое масштабирование

- Данные автоматически распределяются между партициями.
- AWS следит за производительностью и перераспределяет нагрузку при росте запросов.

4. Индексация

- **Primary Key (PK + SK)** — главный индекс, по которому происходит партиционирование.
- **Global Secondary Index (GSI)** — позволяет делать быстрые запросы по другим атрибутам.
- **Local Secondary Index (LSI)** — индекс в рамках одной партиции.

3. Согласованность данных

- Поддерживает **строгую (strong) и итоговую (eventual) согласованность**.
- Итоговая согласованность дешевле и быстрее

5. Транзакции

- Поддерживает **атомарные операции**, обеспечивающие целостность данных.

6. Работа через HTTP API

- Данные передаются в формате JSON.
- Поддерживаются операции `PutItem`, `GetItem`, `Query`, `Scan`, `BatchWriteItem` и другие.

PK и SK в DynamoDB

Primary Key и его роль в партиционировании

DynamoDB использует **Primary Key для распределения данных по партициям**.
Есть два типа первичных ключей:

Простой ключ (Partition Key - PK):

- Каждый элемент имеет **уникальный ключ** (например, `UserID`).
- Партиции формируются на основе хэш-функции от PK.

- Недостаток: нельзя делать поиск по диапазонам значений.
- Пример:

```
{ "UserID": "123", "Name": "Alice", "Age": 25 }
```

Составной ключ (Partition Key + Sort Key — PK + SK):

- Позволяет группировать записи **внутри одной партиции**.
- Используется для хранения связанных данных (например, у актер может учавствовать в нескольких фильмах).
- Позволяет делать запросы типа **"дай все фильмы с актором Tom Hanks"**.
- Пример:

Primary Key		Attributes		
Actor (PARTITION)	**Movie (SORT)**			
Tom Hanks	Cast Away	**Role**	**Year**	**Genre**
		Chuck Noland	2000	Drama
	Toy Story	**Role**	**Year**	**Genre**
		Woody	1995	Children's
Tim Allen	Toy Story	**Role**	**Year**	**Genre**
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	**Role**	**Year**	**Genre**
		Nina Sayers	2010	Drama

Сравнение с классическим SQL

Характеристика	DynamoDB	SQL
Схема	Гибкая, схема не фиксирована	Жёсткая схема с таблицами и связями
Масштабируемость	Автоматическое горизонтальное	Вертикальное и горизонтальное

Характеристика	DynamoDB	SQL
Запросы	Простые, ориентированные на ключи	Сложные SQL-запросы с JOIN и агрегатами
Консистентность	Итоговая или строгая	ACID-комплаентность
Индексация	Первичный ключ + GSI/LSI	Первичный ключ, вторичные индексы
Поддержка транзакций	Да (ограниченная)	Да (полноценная)
Проектирование данных	Основывается на **паттернах доступа**	Можно сначала спроектировать схему, потом писать запросы

Основные функциональные возможности

- **Высокая доступность**: Данные реплицируются автоматически по нескольким регионам AWS.
- **TTL (Time To Live)**: Позволяет автоматически удалять устаревшие записи.
- **Стримы DynamoDB**: Позволяют отслеживать изменения в данных и интегрироваться с AWS Lambda для обработки событий.
- **Автоматическое шифрование**: Данные могут быть зашифрованы с использованием AWS Key Management Service (KMS).
- В отличие от SQL, где сложные аналитические запросы выполняются с использованием JOIN и GROUP BY, в DynamoDB необходимо проектировать схему данных с учетом специфики запросов.
- Вместо традиционной поддержки ACID-транзакций DynamoDB использует упрощённый механизм с ограничениями по размеру транзакции.

Вот дополненный текст с первым пунктом про HTTP-доступ и информацией об авторизации через AWS IAM:

Возможность интеграции со сторонними приложениями

1. Прямые HTTP-запросы (AWS API Gateway + IAM Auth)

DynamoDB позволяет выполнять запросы напрямую через HTTP, передавая **заголовки авторизации AWS** и JSON-пейлоад.

- **Пример HTTP-запроса на `PutItem`:**

```
POST https://dynamodb.us-east-1.amazonaws.com/
Content-Type: application/x-amz-json-1.0
X-Amz-Target: DynamoDB_20120810.PutItem
Authorization: AWS4-HMAC-SHA256 Credential=..., Signature=...

{
  "TableName": "Rentals",
  "Item": {
    "PK": {"S": "CUSTOMER#005"},
    "SK": {"S": "PROFILE"},
    "Name": {"S": "Козлов"},
    "Discount": {"N": "4"},
    "Region": {"S": "Нижегородский"}
  }
}
```

---

2. AWS SDK (Go, Python, Java, Node.js и др.)

AWS предоставляет **официальные SDK** для популярных языков программирования.
Это **основной способ** работы с DynamoDB в коде.

- **Пример использования Go SDK:**

```go
sess := session.Must(session.NewSession(&aws.Config{
    Region: aws.String("us-east-1"),
}))

svc := dynamodb.New(sess)

input := &dynamodb.GetItemInput{
    TableName: aws.String("Rentals"),
    Key: map[string]*dynamodb.AttributeValue{
        "PK": {S: aws.String("CUSTOMER#002")},
        "SK": {S: aws.String("PROFILE")},
    },
}

result, err := svc.GetItem(input)
if err != nil {
    log.Fatalf("Failed to get item: %v", err)
}

fmt.Println(result.Item)
```

---

3. AWS CLI (Командная строка)

AWS CLI позволяет управлять DynamoDB без кода, через терминал.

Пример добавления записи:

```
aws dynamodb put-item --table-name Rentals --item '{
  "PK": {"S": "CUSTOMER#005"},
  "SK": {"S": "PROFILE"},
```

```
    "Name": {"S": "Козлов"},
    "Discount": {"N": "4"},
    "Region": {"S": "Нижегородский"}
}'
```

Так же нужна авторизация

```
aws configure
```

---

4. NoSQL Workbench (официальный GUI от AWS)

AWS NoSQL Workbench — это **официальное GUI-приложение** для работы с DynamoDB.

- Проектирование схемы базы
- Выполнение запросов (`GetItem` , `Query` , `Scan` и т. д.)
- Визуализация данных

---

5. DynamoDB Admin (неофициальный, но удобный GUI)

DynamoDB Admin — это **open-source GUI**, который можно запустить локально в Docker.

📌 **Запуск через Docker:**

```
docker run -p 8001:8001 -e DYNAMO_ENDPOINT=http://localhost:8000 aaronshaf/dynamodb-admin
```

✓ Просмотр и редактирование данных
✓ Выполнение запросов
✓ Удобный интерфейс для работы с DynamoDB
