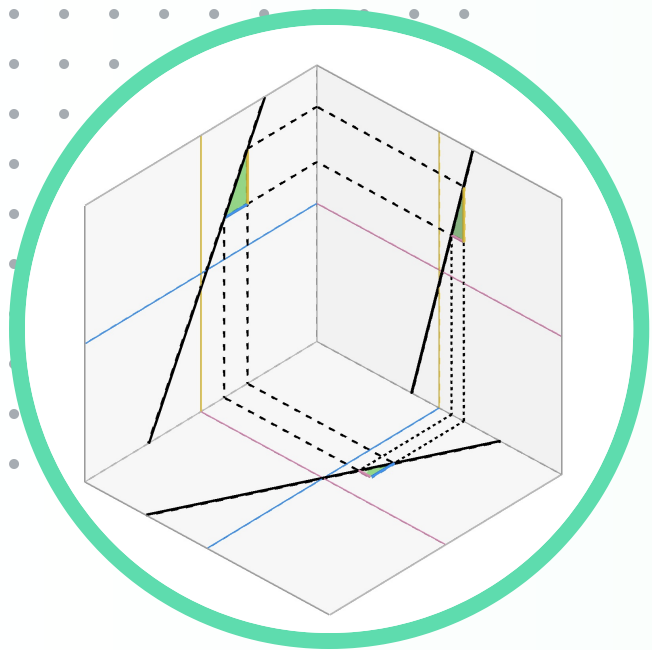


自己紹介



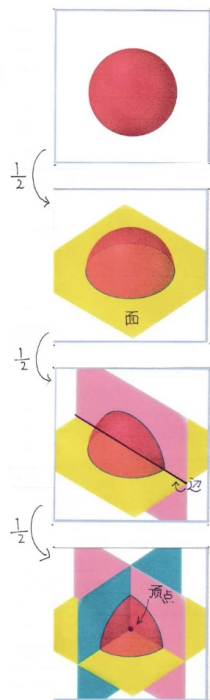


tomixy (菅野 亜実)

2002年生まれ => 現在20歳

教材制作 (GeoGebra・LaTeX)

☆ 単位格子(立方体)内の原子(球)の数の数え方



空間内の球は 1 個

面上の球は $\frac{1}{2}$ 個が立方体内にある

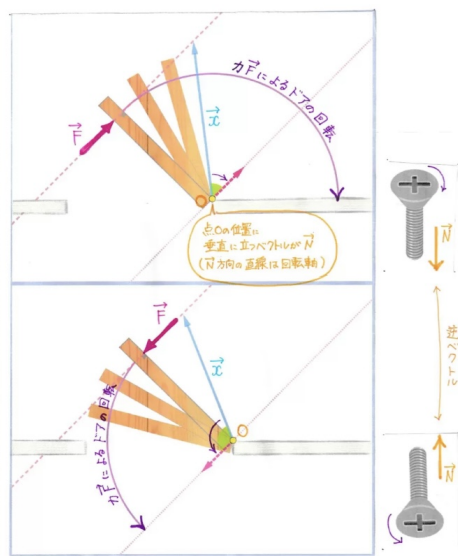
辺上の球は $\frac{1}{4}$ 個が立方体内にある

頂点を含む球は $\frac{1}{8}$ 個が立方体内にある

< def. 力のモーメント >

点OからFの作用線へのベクトルを \vec{x} とすると,

$$\text{点Oのまわりの力のモーメント } \vec{N} \stackrel{\text{def}}{=} \vec{x} \times \vec{F}$$



ネジの進行方向による外積の定義のおかげで

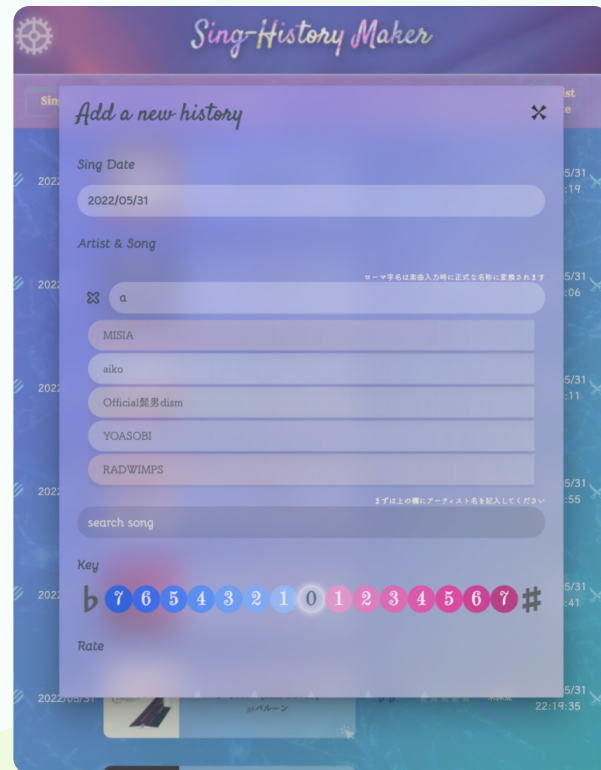
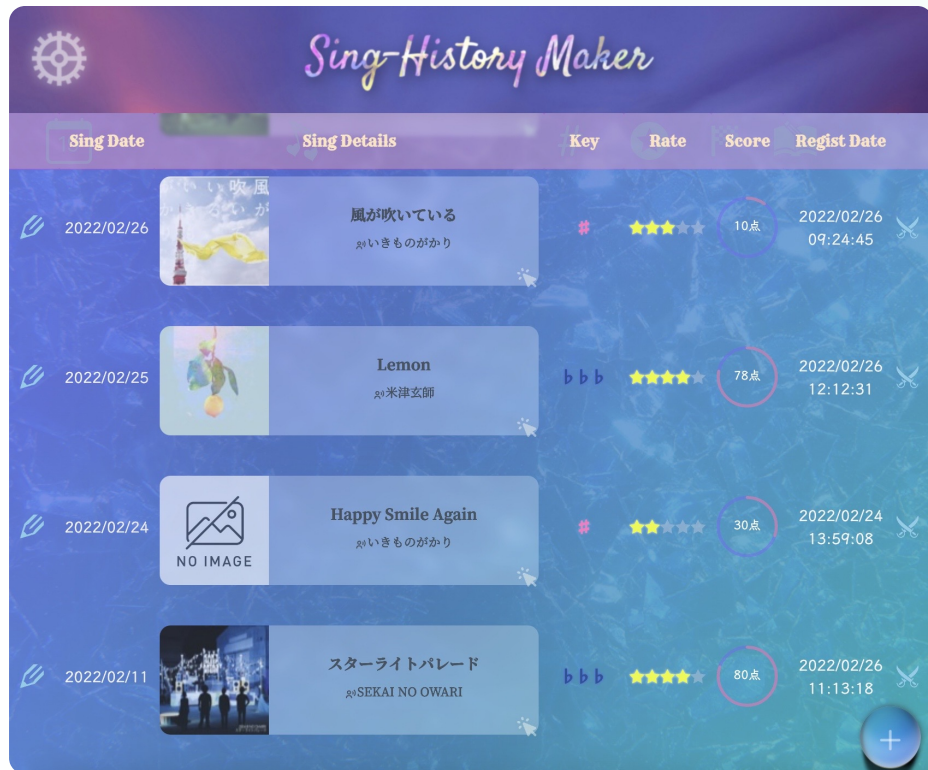
逆回転を表す力のモーメントは 逆ベクトル として扱うことができる

1 カルボン酸


1.1 カルボン酸 (1) 類の端 CH_3 を **カルボキシル基** $-\text{COOH}$ に置き換えた化合物 **-oic acid**

$\text{H}-\text{C}(=\text{O})-\text{OH}$	$\text{H}-\text{C}(=\text{O})-\text{OH}$ COOH の C の位置番号を最小にする。 メタン酸 methanoic acid ...methan o ic (慣用名 ギ酸) formic acid
$\text{H}_3\text{C}-\text{C}(=\text{O})-\text{OH}$	$\text{H}_3\text{C}-\text{C}(=\text{O})-\text{OH}$ ethanoic acid ...ethan o ic (慣用名 酢酸) acetic acid

カラオケ記録アプリ (React)



hereafter = 教材開発 + フロントエンド開発



イメージでわかる ページネーションAPI

GraphQLとREST

artistNameとsongNameだけ欲しい…

id	5
artistId	270311351
artistName	UNISON SQUARE GARDEN
songName	harmonized finale
jacketUrl	270311351_harmonized-finale.jpg
rating	4
score	86

REST API Request

```
https://ex.app/api/song?id=5&fields=artistName,songName
```

GraphQL Request

```
{  
  song(id: 5) {  
    artistName  
    songName  
  }  
}
```

Response

```
{  
  "data": {  
    "song": {  
      "artistName": "UNISON SQUARE GARDEN",  
      "songName": "harmonized finale"  
    }  
  }  
}
```

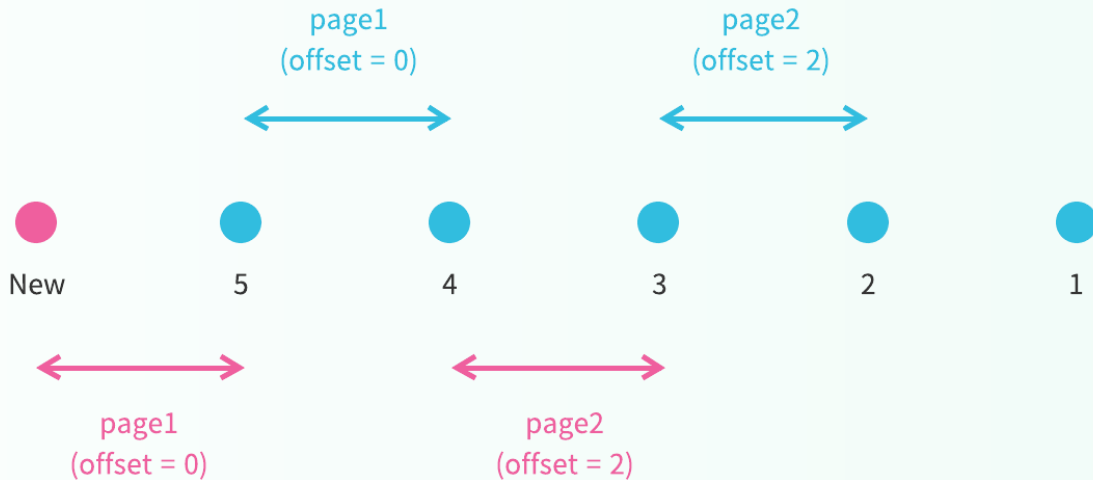
Offset Pagination

読み飛ばす数（相対位置）で管理

👍 任意のページのデータを得るクエリ（エンドポイント）を把握できる

👎 新たに追加されたデータがあればそのデータの個数分ずれる

→ ページ送り（追加頻度が少なく、位置を覚えやすい場合）



Cursor Pagination

「このデータまでは読んだ」という一意な栞（絶対位置）を決め、次のリクエストを決めるためのメタ情報を渡す

- どこからどこまで読んだか (`startCursor`と`endCursor`)
- 次のページはあるか (`hasNextPage`)
- 前のページはあるか (`hasPreviousPage`)

👍 新たに追加されたデータがあっても正しく続きを取得できる

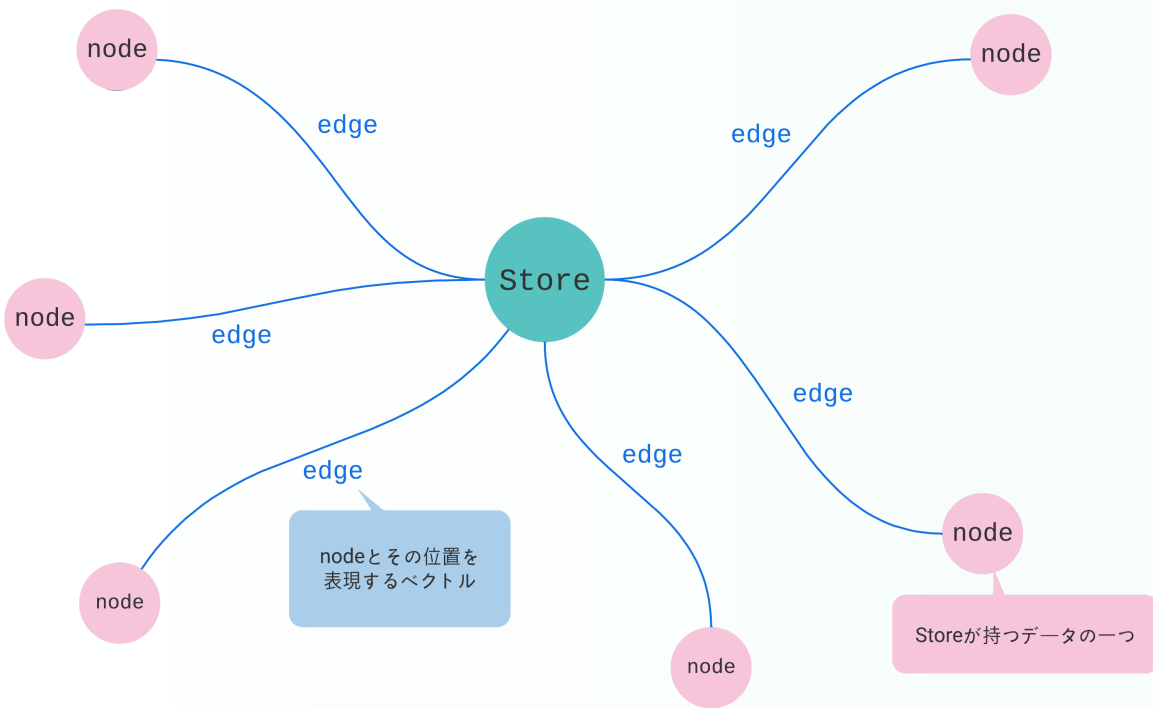
🗨️ 直前のページ、直後のページの情報しかわからない

→ 無限スクロール、Moreボタン（頻繁に追加され位置が変動する場合）

RelayCursor式のResponse

~ Thinking in Graphs ~

edgesとpageInfo



```
{
  "data": {
    "setlistPerPage": {
      "edges": [
        {
          "cursor": "4",
          "node": {
            "artistName": "Whiteberry",
            "songName": "夏祭り"
          }
        },
        {
          "cursor": "5",
          "node": {
            "artistName": "UNISON SQUARE GARDEN",
            "songName": "harmonized finale"
          }
        }
      ],
      "pageInfo": {
        "startCursor": "4",
        "endCursor": "5",
        "hasNextPage": true,
        "hasPreviousPage": false
      }
    }
  }
}
```

RelayCursor式のRequest

取得開始場所

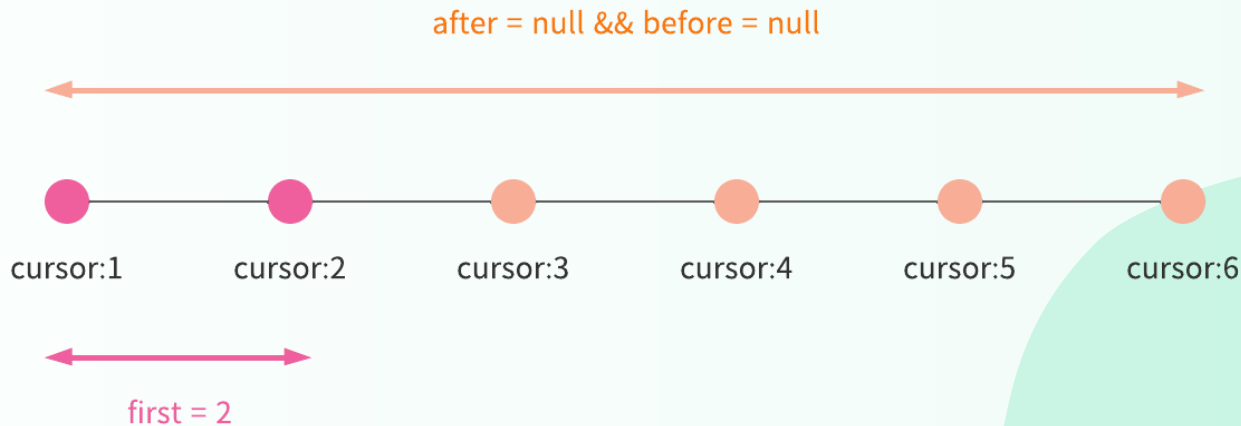
before | after

取得する数

first | last

get First Page

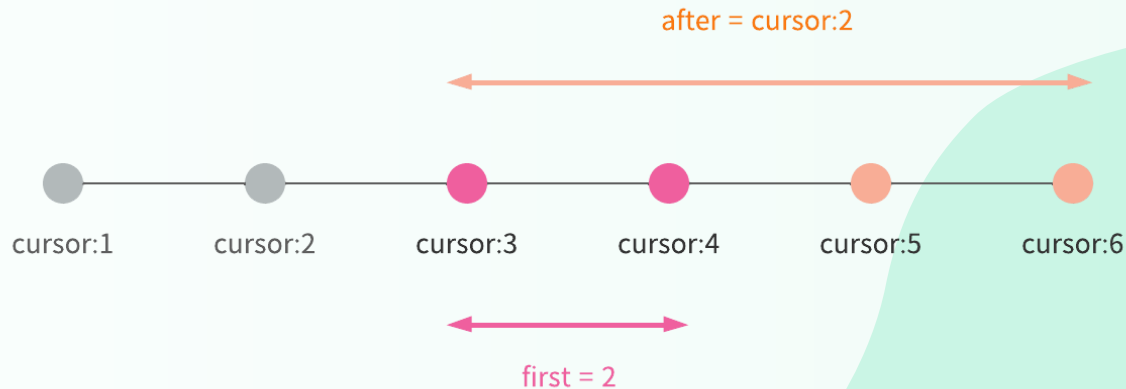
```
{
  setlistPerPage(first: 2) {
    edges {
      cursor
      node {
        artistName
        songName
      }
    }
    pageInfo {
      startCursor
      endCursor
      hasNextPage
      hasPreviousPage
    }
  }
}
```



get Next Page

after === 直前のページのendCursor

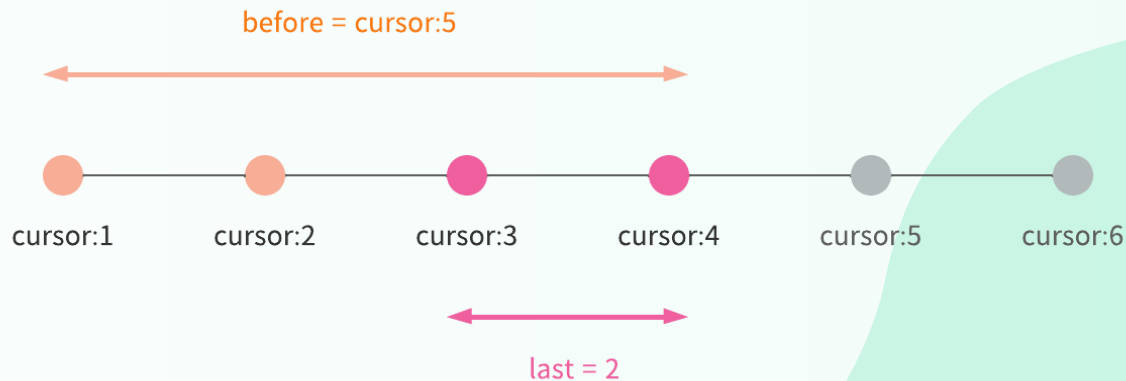
```
{
  setlistPerPage(first: 2, after: "cursor:2") {
    edges {
      cursor
      node {
        artistName
        songName
      }
    }
  }
  pageInfo {
    startCursor
    endCursor
    hasNextPage
    hasPreviousPage
  }
}
```



get Previous Page

before === 直後のページのstartCursor

```
{
  setlistPerPage(last: 2, before: "cursor:6") {
    edges {
      cursor
      node {
        artistName
        songName
      }
    }
  }
  pageInfo {
    startCursor
    endCursor
    hasNextPage
    hasPreviousPage
  }
}
```



改めてページネーションAPIを比較

リアルタイム性とページネーション

Offset Pagination

読み飛ばす数（相対位置）で管理

👉 任意のページのデータを得るクエリ（エンドポイント）を把握できる

👈 新たに追加されたデータがあればそのデータの個数分ずれる

→ ページ送り（追加頻度が少なく、位置を覚えやすい場合）

Cursor Pagination

「このデータまでは読んだ」という一意な栞（絶対位置）で管理

👉 新たに追加されたデータがあっても正しく続きを取得できる

👈 直前のページ、直後のページの情報しかわからない

→ 無限スクロール、Moreボタン（頻繁に追加され位置が変動する場合）

Usability is derived from specifications

ユーザビリティは仕様から導く