Normalisation

▼ What is normalisation?

process of organsing data to minimise redundancy and improve data integrity. It is widely used in databases to structure the data logically and efficiently.

We flatten the data structure, store these entities separately and also ensure relationships between various entities using unique IDs.

▼ What are the benefits of normalisation?

Remove Redundant Data: Ensure that the same piece of data is not stored in multiple places.

Increase Efficiency: Organize the data to make it easy to retrieve and update without inconsistency.

Simplify nested relationships

Store related data together and unrelated data separately.

▼ Example 1

Example Data

```
// Problem statement:

const state = {
  users: [
    {
      id: 1,
      name: "Alice",
      posts: [
        { id: 101, title: "Post 1" },
        { id: 102, title: "Post 2" },
      ],
```

```
    },
    {
        id: 2,
        name: "Bob",
        posts: [
            { id: 103, title: "Post 3" }
        ]
    },
  ],
};
```

Normalised Data

1. Store the users and posts separately
2. Store only the ids of the posts in the users data
3. Store all the users and posts with respect to IDs and maintain a list of all IDs for each entities (i.e. for both users and posts)

```
// Normalized Data

const state = {
    users: {
        byIds: {
            1: {
                id: 1,
                name: "Alice",
              },
            2: {
                id: 2,
                name: "Bob",
            }
        },
        allIds: [1, 2]
    },
    posts: {
```

```
        byIds: {
            101: { id: 101, title: "Post 1", userId: 1 },
            102: { id: 102, title: "Post 2", userId: 1 },
            103: { id: 103, title: "Post 3", userId: 2 }
        },
        allIds: [101, 103, 102]
    },
}

state.users.byIds['2'] // O(1) Increase efficiency of lookup
```

▼ **Example 2**

Example Data:

```
const state = {
    users: [
      { id: 1, name: 'Alice', posts: [ { id: 101, title: 'Pos
      { id: 2, name: 'Bob', posts: [ { id: 102, title: 'Post
    ],
    tags: [
      { id: 301, name: 'Tech', posts: [ { id: 101 }, { id: 1(
      { id: 302, name: 'Travel', posts: [ { id: 102 } ] },
    ],
  };
```

Normalised Data

1. Store the users, posts, comments and tags separately
2. Store only the ids for every relation which exists in the data (i.e. only the ids of posts in users data, ids of posts in tags data, ids of comments in post data)
3. Store all the users, posts, comments and tags with respect to IDs and maintain a list of all IDs for each entities (i.e. for both users and posts)

```
const normalized_state = {
  users: {
      byIds: {
          1: { id: 1, name: 'Alice', posts: [ 101 ] },
          2: { id: 2, name: 'Bob', posts: [ 102] },
      },
      allIds: [1, 2]
  },
  posts: {
      byIds: {
          101: { id: 101, title: 'Post 1', comments: [201]
          102: { id: 102, title: 'Post 2', comments: [202]
      },
      allIds: [101, 102]
  },
  comments: {
      byIds: {
          201: { id: 201, text: 'Comment 1' },
          202: { id: 202, text: 'Comment 2' }
      },
      allIds: [201, 202]
  },
  tags: {
      byIds: {
          301: { id: 301, name: 'Tech', posts: [ 101, 102
          302: { id: 302, name: 'Travel', posts: [ 102 ] }
      },
      allIds: [301, 302]
  }
}
```