Referensi Cepat GraphQL

Schema, query, mutation, tipe, fragment

Definisi Schema

Root Schema

```
schema {
  query: Query
  mutation: Mutation
}
```

Object Type

```
type User {
  id: ID!
  name: String!
  email: String
}
```

Query

Query Dasar

```
query {
  user(id: "1") {
    name
    email
  }
}
```

Query Bernama dengan Alias

```
query GetUsers {
  admin: user(role: ADMIN) { name }
  guest: user(role: GUEST) { name }
}
```

Mutation

Mutation Dasar

```
mutation {
  createUser(input: { name: "Alice" }) {
    id
    name
  }
}
```

Tipe Input

```
input CreateUserInput {
  name: String!
  email: String
}
```

Subscription

Subscription Dasar

```
subscription {
  messageAdded(channel: "general") {
    text
    sender { name }
  }
}
```

Ringkasan

`subscription`	Data real-time via WebSocket
`Server push`	Server mengirim pembaruan ke client
`Single field`	Hanya satu root field per subscription

Tipe

Scalar Type

`Int`	Integer 32-bit bertanda
`Float`	Bilangan pecahan presisi ganda
`String`	Urutan karakter UTF-8
`Boolean`	true atau false
`ID`	Identifier unik (diserialisasi sebagai String)

Modifier Tipe

`String`	String nullable
`String!`	String non-null
`[String]`	List nullable berisi string nullable
`[String!]!`	List non-null berisi string non-null

Enum & Union

```
enum Role { ADMIN USER GUEST }
union SearchResult = User | Post
interface Node { id: ID! }
```

Argumen & Variabel

Variabel

```
query GetUser($id: ID!) {
  user(id: $id) {
    name
  }
}
# Variables: { "id": "123" }
```

Nilai Default

```
query GetUsers($limit: Int = 10) {
  users(limit: $limit) { name }
}
```

Fragment

Fragment Bernama

```
fragment UserFields on User {
  id
  name
  email
}
```

Menggunakan Fragment

```
query {
  user(id: "1") { ...UserFields }
  admin: user(id: "2") { ...UserFields }
}
```

Inline Fragment

```
query {
  search(text: "a") {
    ... on User { name }
    ... on Post { title }
  }
}
```

Directive

Directive Bawaan

`@include(if: Boolean!)`	Sertakan field hanya jika kondisi benar
`@skip(if: Boolean!)`	Lewati field jika kondisi benar
`@deprecated(reason: String)`	Tandai field sebagai deprecated

Contoh Penggunaan

```
query GetUser($withEmail: Boolean!) {
  user(id: "1") {
    name
    email @include(if: $withEmail)
  }
}
```

Introspection

Introspection Tipe

```
query {
  __type(name: "User") {
    name
    fields { name type { name } }
  }
}
```

Introspection Schema

```
query {
  __schema {
    types { name kind }
    queryType { name }
  }
}
```

Field Introspection

`__schema`	Query tipe dan directive dari schema
`__type(name:)`	Query tipe tertentu berdasarkan nama
`__typename`	Mengembalikan nama tipe dari objek mana pun

Pola Umum

Pagination (gaya Relay)

```
query {
  users(first: 10, after: "cursor") {
    edges { node { name } cursor }
    pageInfo { hasNextPage }
  }
}
```

Penanganan Error

```
{
  "data": { "user": null },
  "errors": [{ "message": "Not found",
    "path": ["user"] }]
}
```

Praktik Terbaik

`Beri nama operasi`	Selalu beri nama query dan mutation
`Gunakan variabel`	Jangan interpolasi nilai langsung ke query
`Minta field yang dibutuhkan`	Hindari over-fetching dengan seleksi yang tepat
`Gunakan fragment`	Bagikan kumpulan field antar query