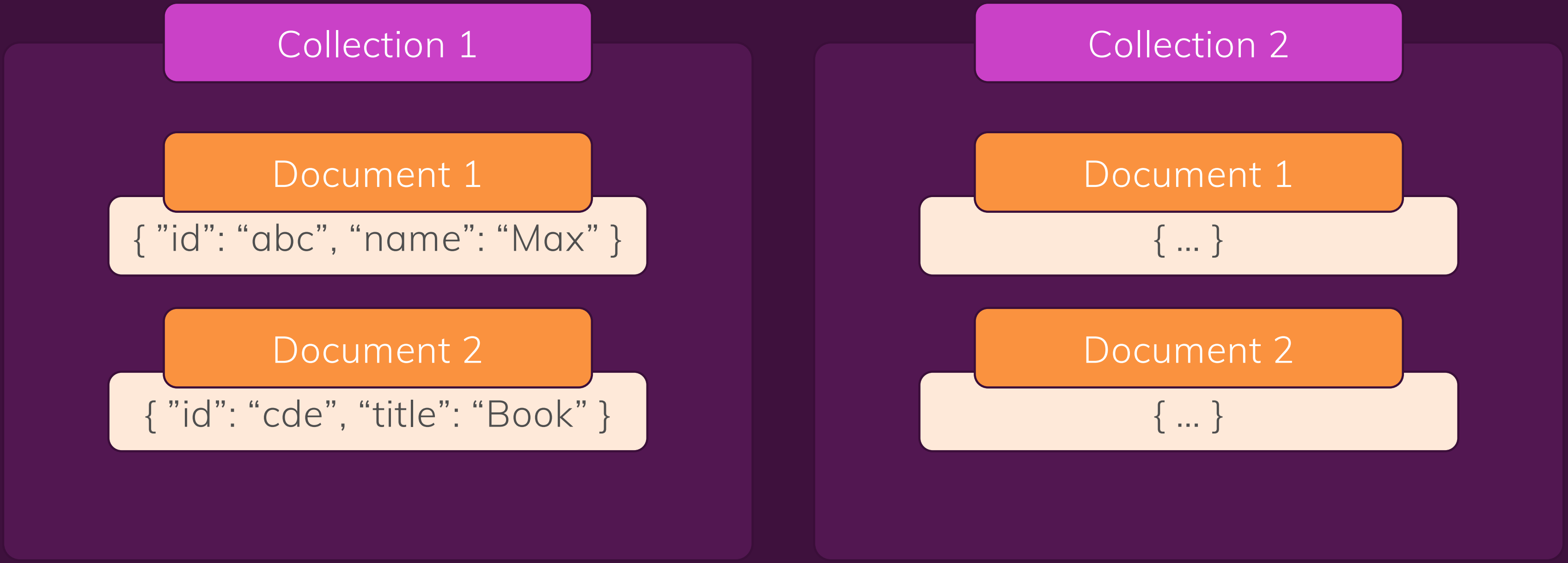
The Idea Behind NoSQL Database Systems

Store data without focusing on a strict schemas / data structures or relationships across multiple tables



Collection 1

Document 1

{ ”id”: “abc”, “name”: “Max” }

Document 2

{ ”id”: “cde”, “title”: “Book” }

Collection 2

Document 1

{ … }

Document 2

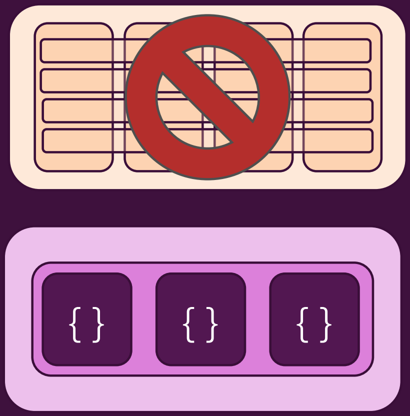
{ … }

Understanding Collections & Documents

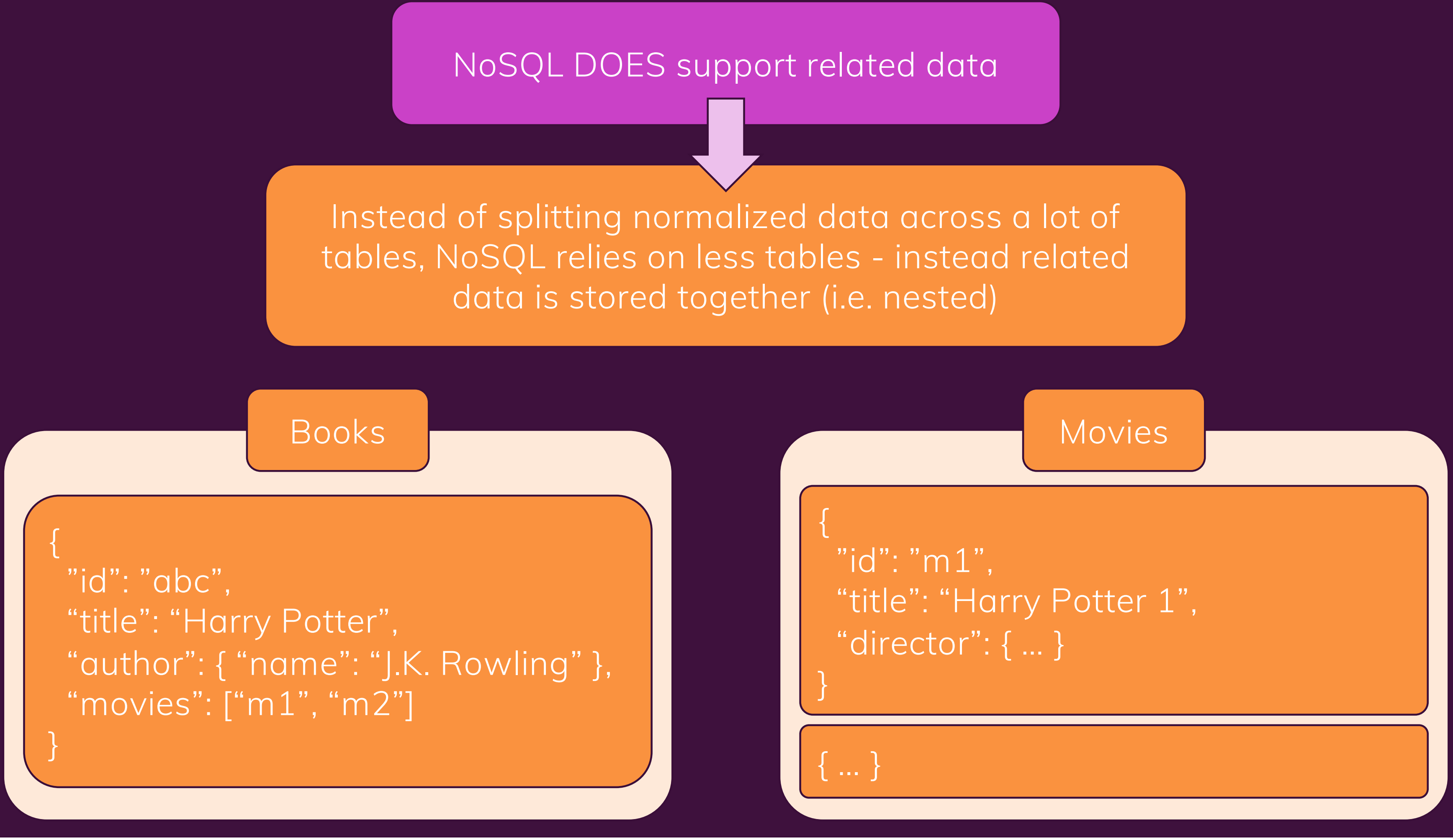
“Collections” are a bit like “Tables” in SQL databases but they don’t contain “columns” and “rows”

Instead, “Collections” contain “Documents”

“Documents” are like JavaScript objects – complex data structures with key-value pairs



NoSQL & Related Data



NoSQL DOES support related data

Instead of splitting normalized data across a lot of tables, NoSQL relies on less tables - instead related data is stored together (i.e. nested)

Books

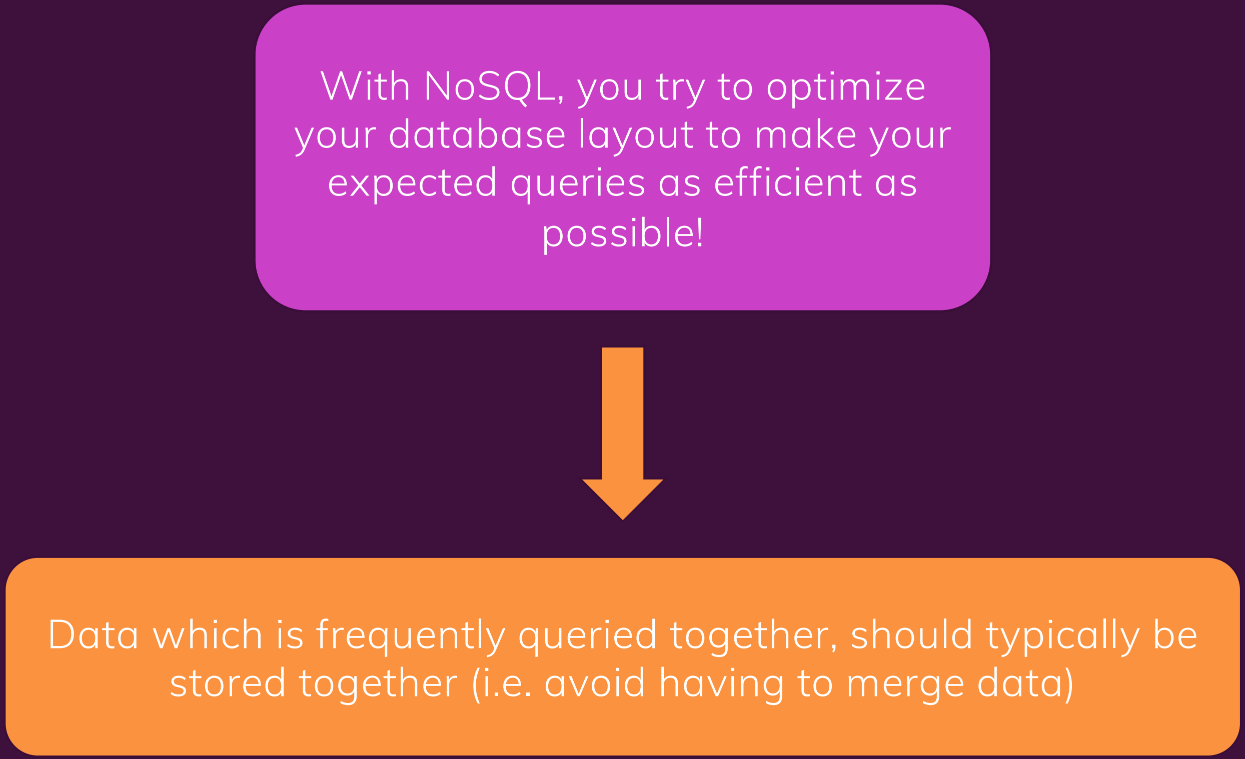
```
{
  ”id”: ”abc”,
  “title”: “Harry Potter”,
  “author”: { “name”: “J.K. Rowling” },
  “movies”: [“m1”, “m2”]
}
```

Movies

```
{
  ”id”: ”m1”,
  “title”: “Harry Potter 1”,
  “director”: { … }
}
```

```
{ … }
```

Plan Your Database Queries!

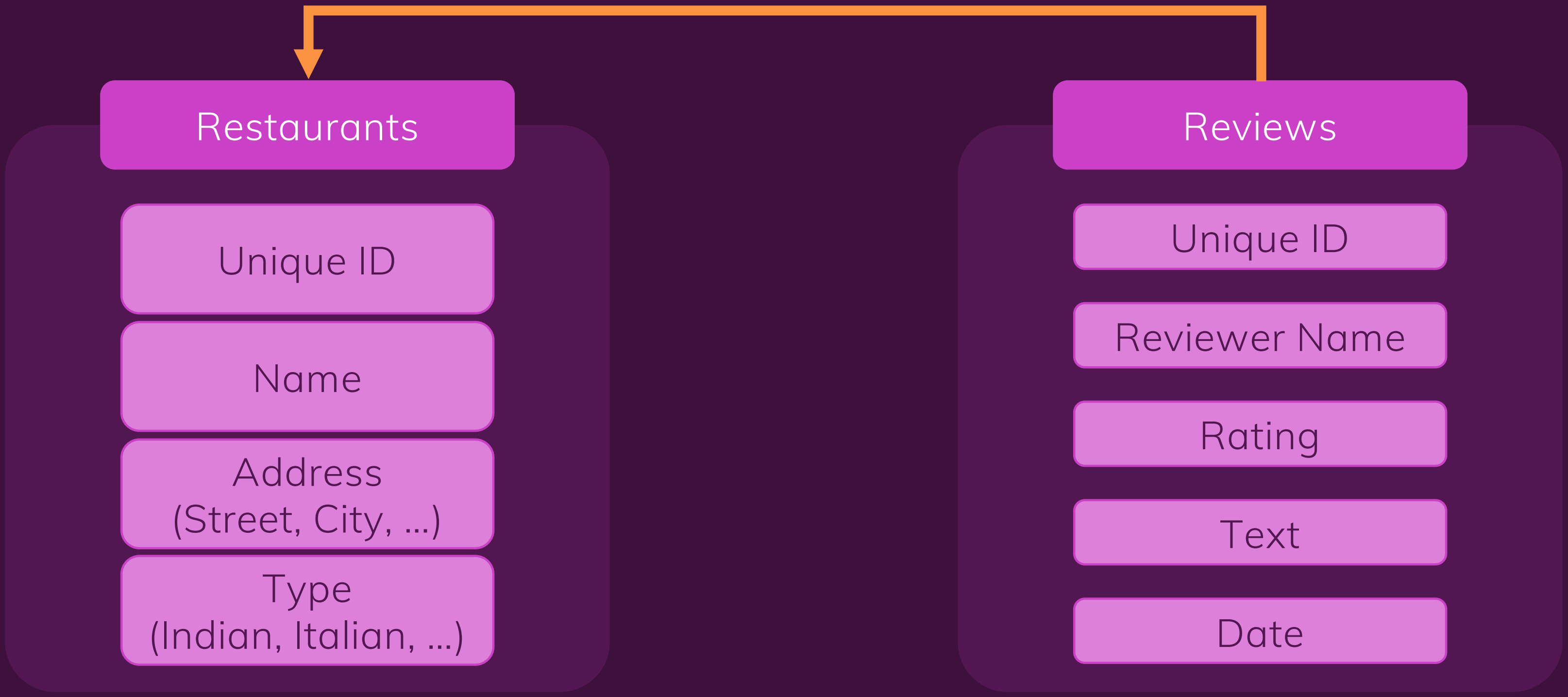


With NoSQL, you try to optimize your database layout to make your expected queries as efficient as possible!

Data which is frequently queried together, should typically be stored together (i.e. avoid having to merge data)

ACADEMIND

An Example



Restaurants

Unique ID

Name

Address
(Street, City, …)

Type
(Indian, Italian, …)

Reviews

Unique ID

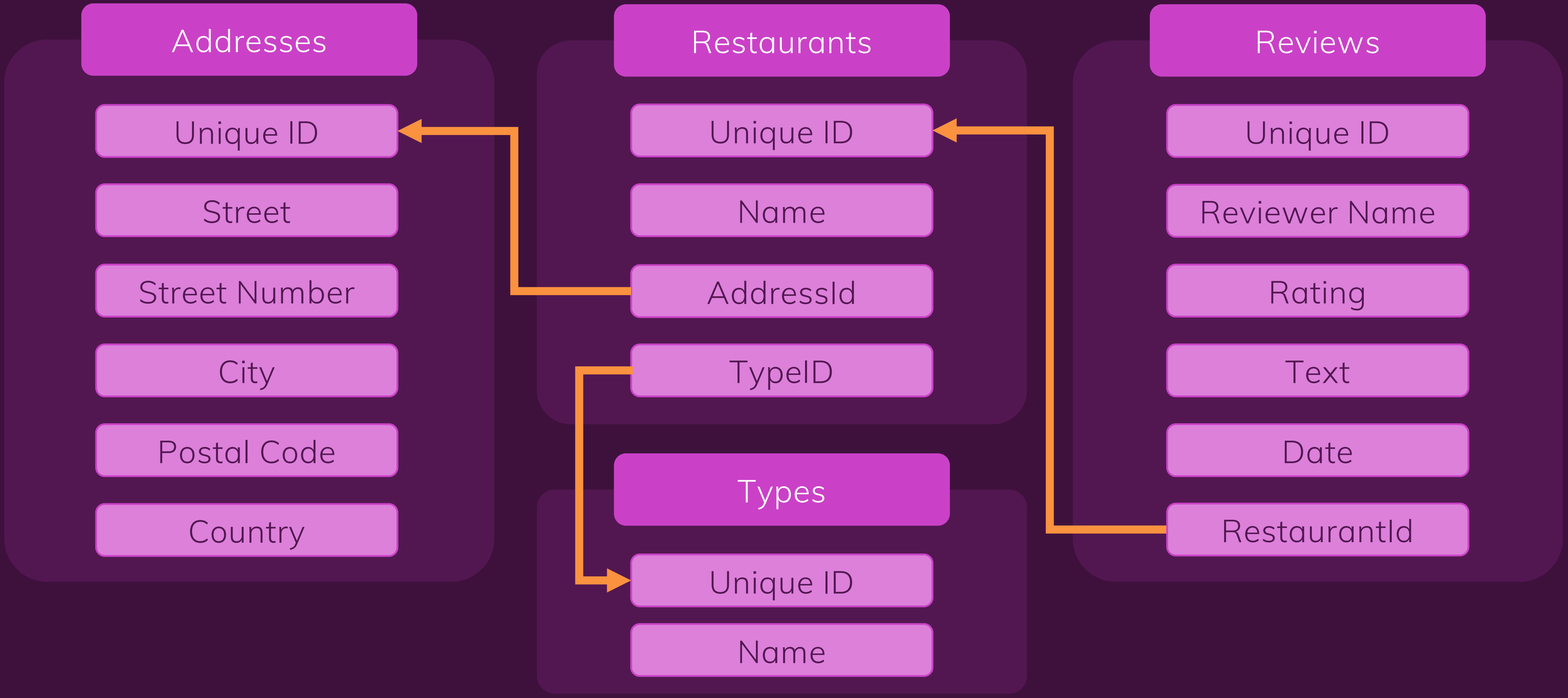
Reviewer Name

Rating

Text

Date

SQL World



Addresses

- Unique ID
- Street
- Street Number
- City
- Postal Code
- Country

Restaurants

- Unique ID
- Name
- AddressId
- TypeID

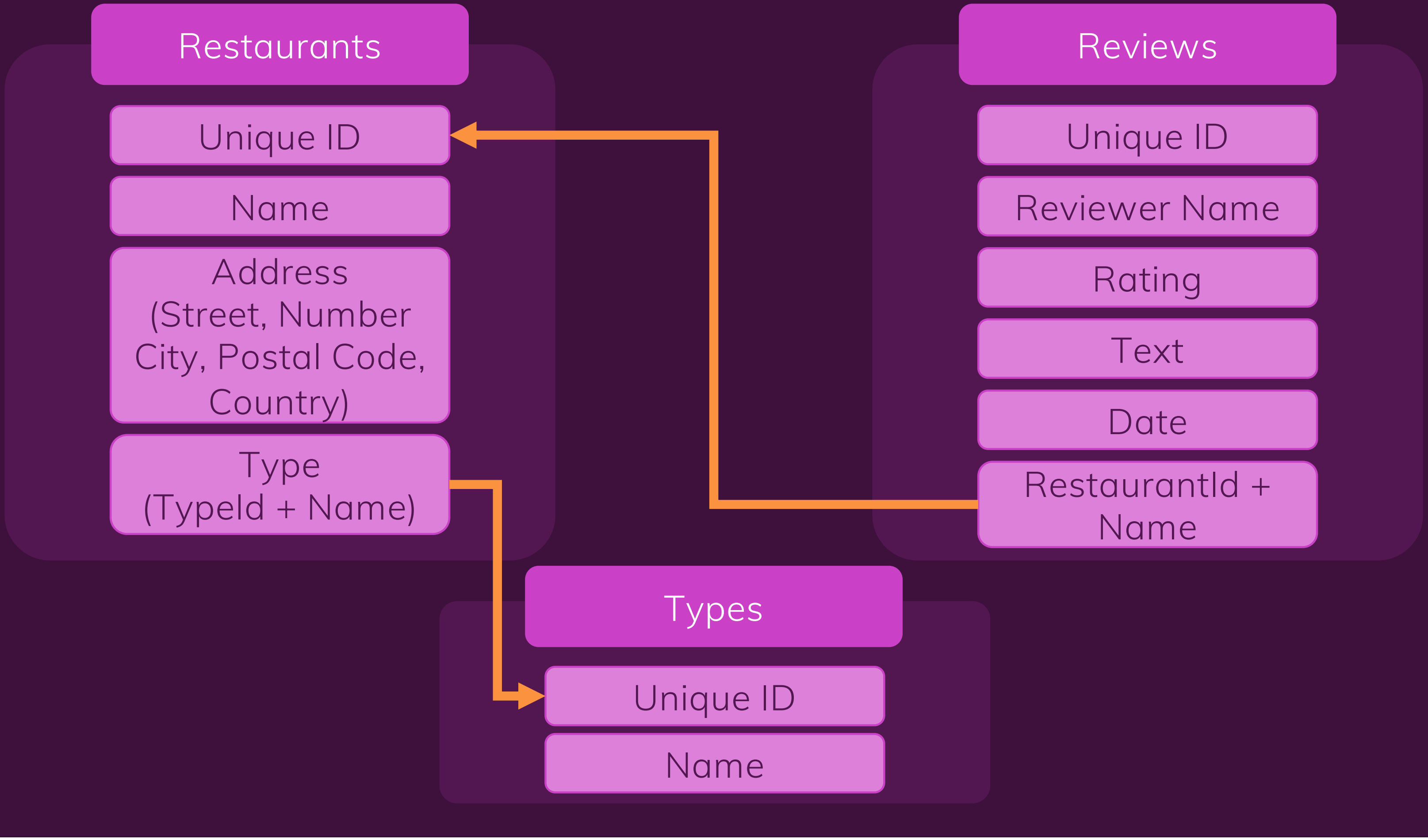
Types

- Unique ID
- Name

Reviews

- Unique ID
- Reviewer Name
- Rating
- Text
- Date
- RestaurantId

A Possible NoSQL Structure



Restaurants

- Unique ID
- Name
- Address (Street, Number City, Postal Code, Country)
- Type (TypeId + Name)

Reviews

- Unique ID
- Reviewer Name
- Rating
- Text
- Date
- RestaurantId + Name

Types

- Unique ID
- Name