DANIEL GRAHAM

FIRESTORE LECTURE II

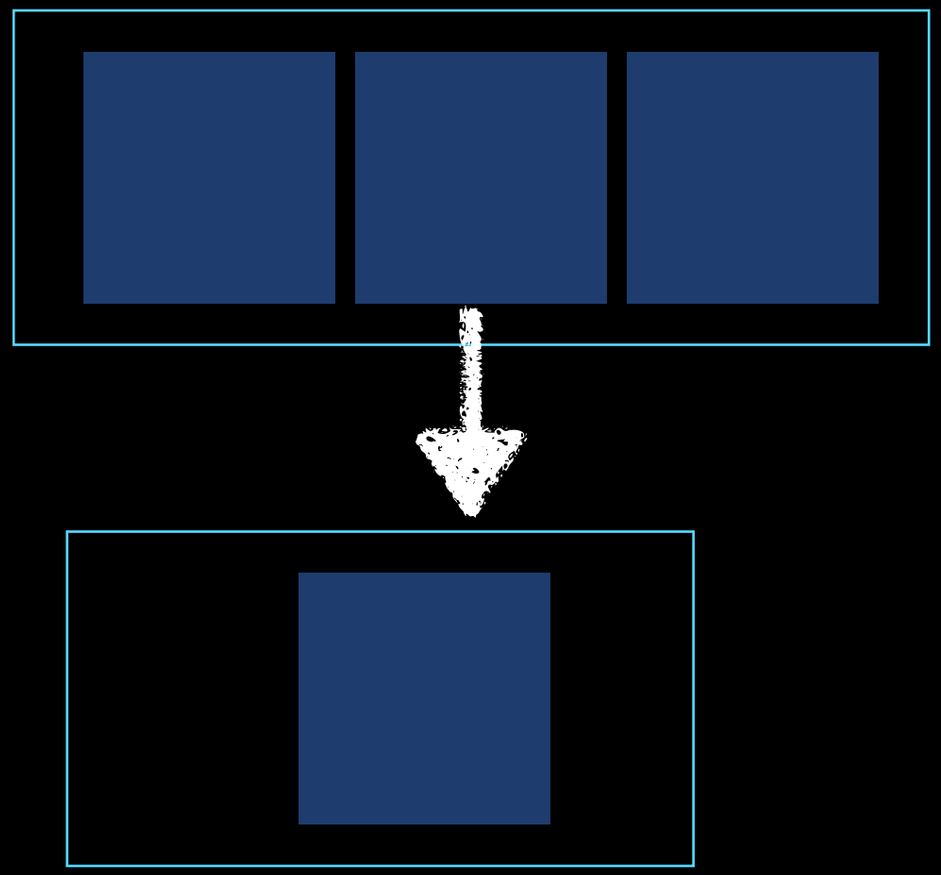
OVERVIEW

- Querying
- Security

QUERYING

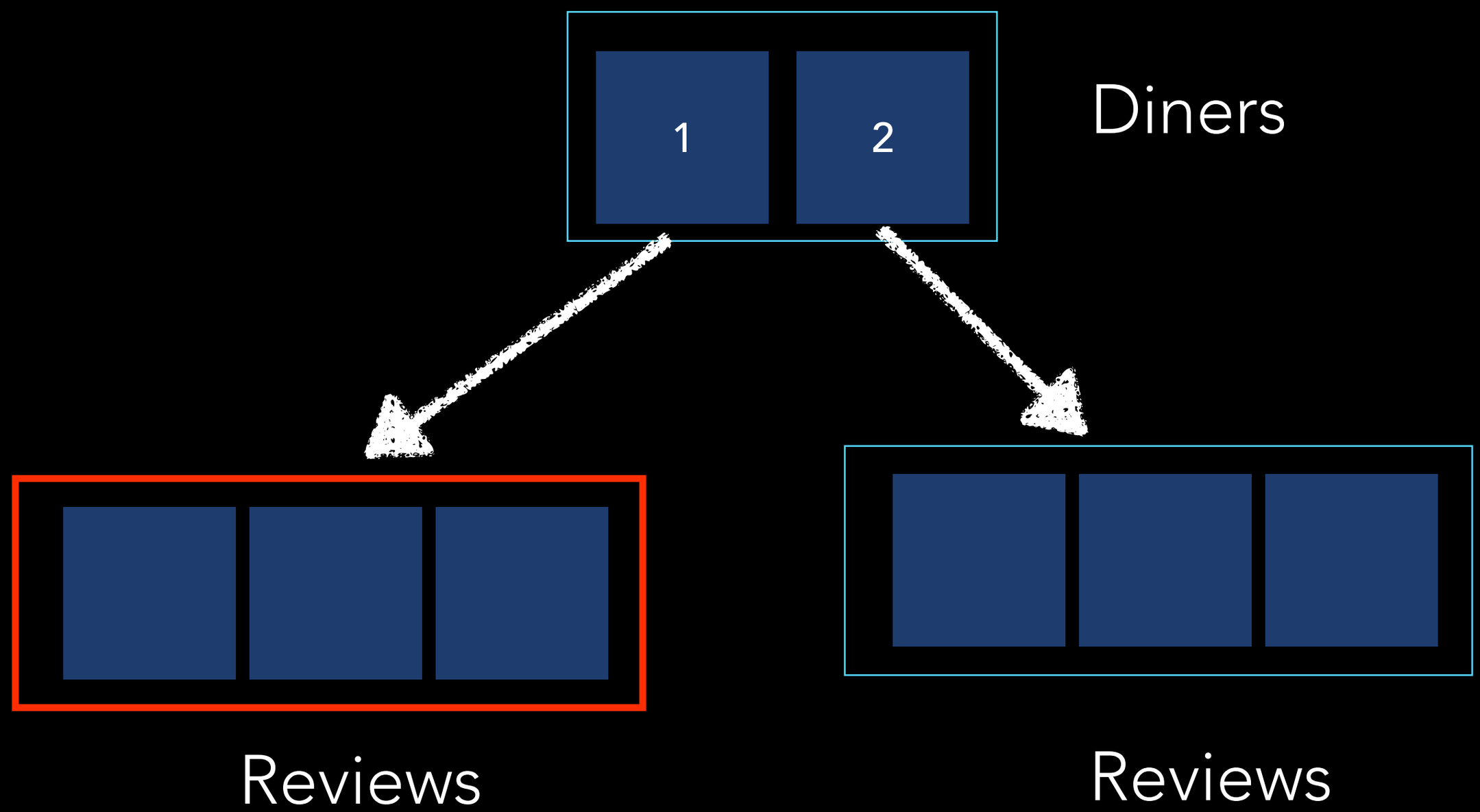
- The process of finding something in a database

RULE: YOU CAN ONLY QUERY COLLECTIONS/SUB-COLLECTIONS

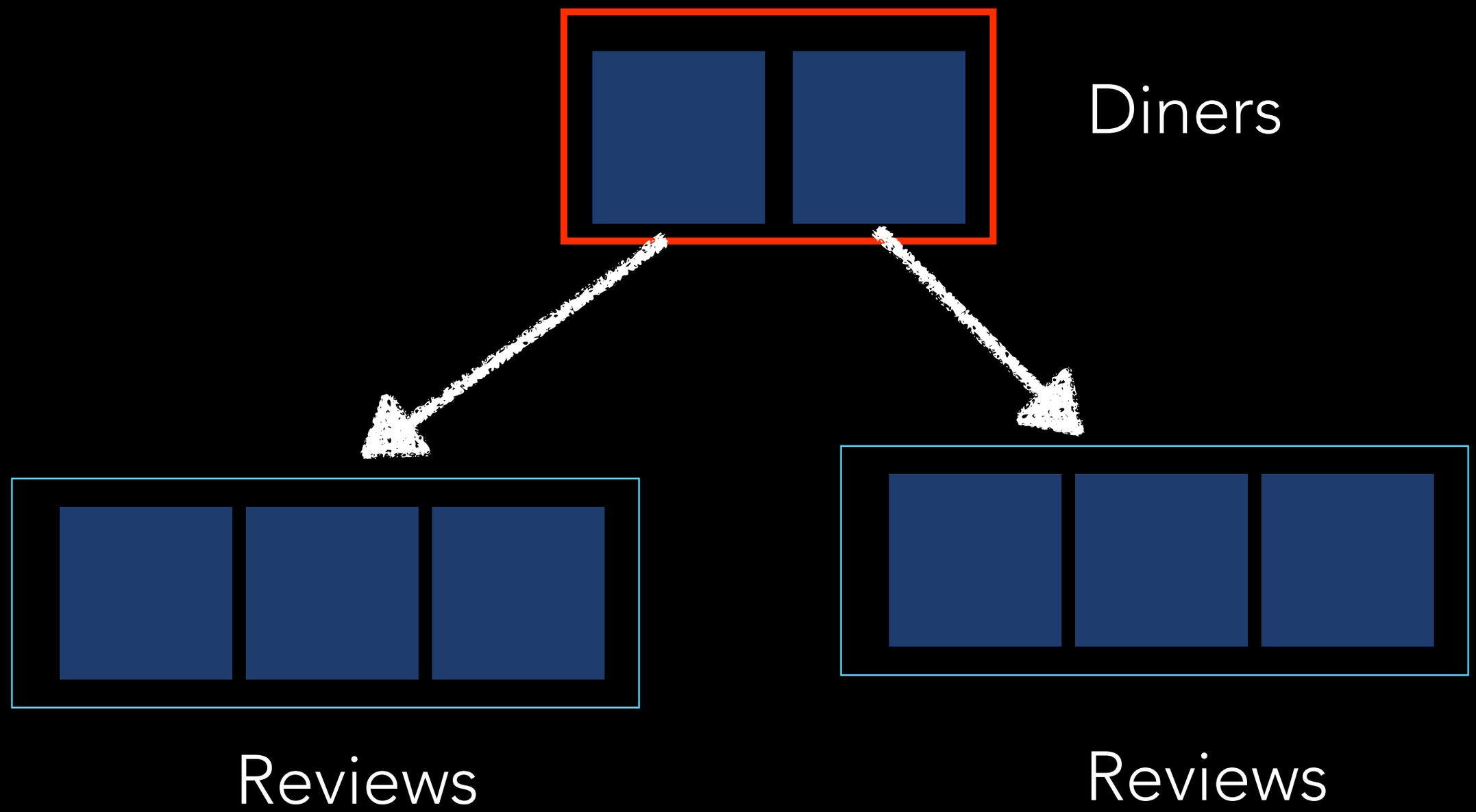


Query: Find 4 star reviews for dinner 1

Possible



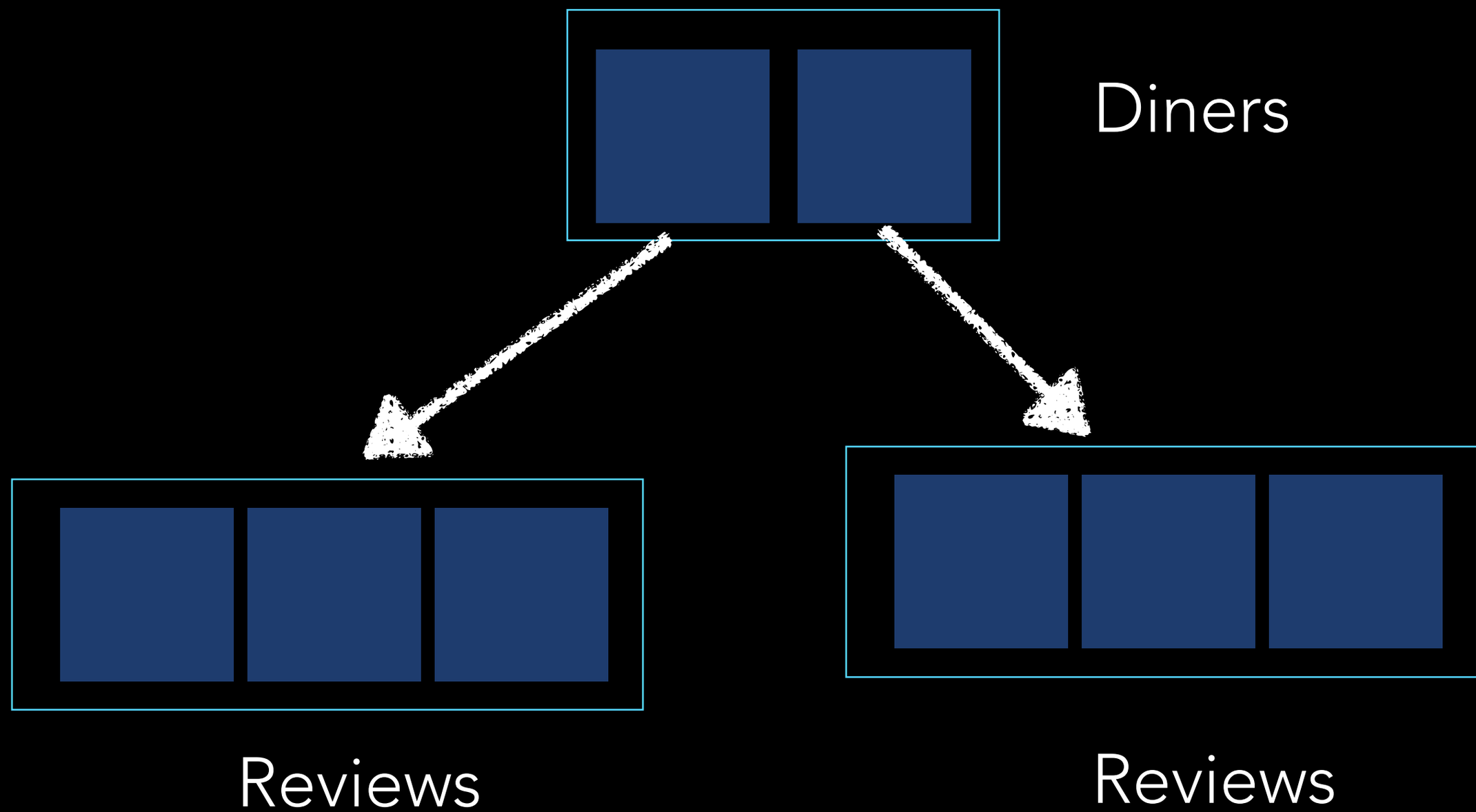
Find all Diner In the north



Collection Group Query

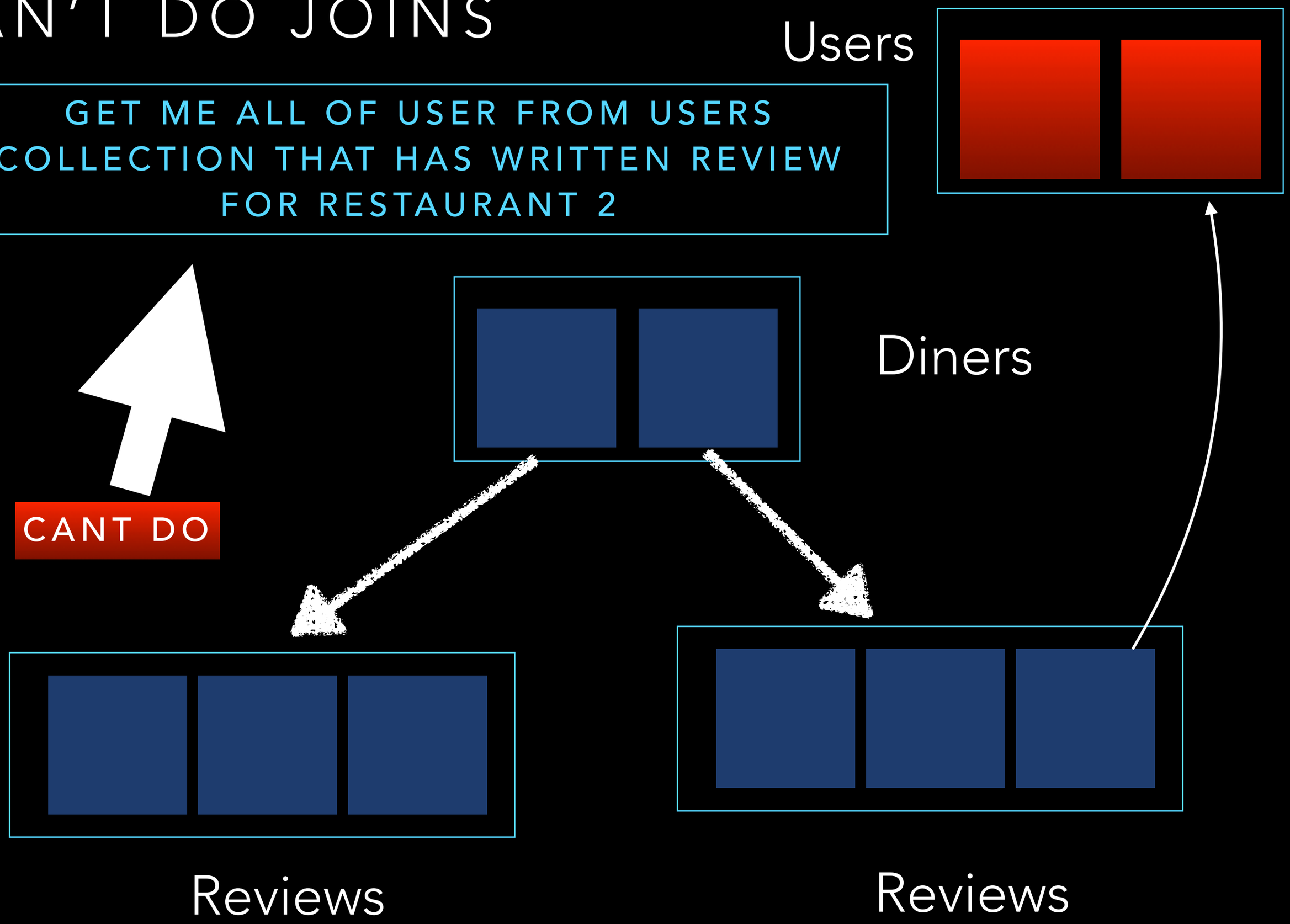
Queries that span multiple

YOU WILL HAVE MANUAL INDEX



CAN'T DO JOINS

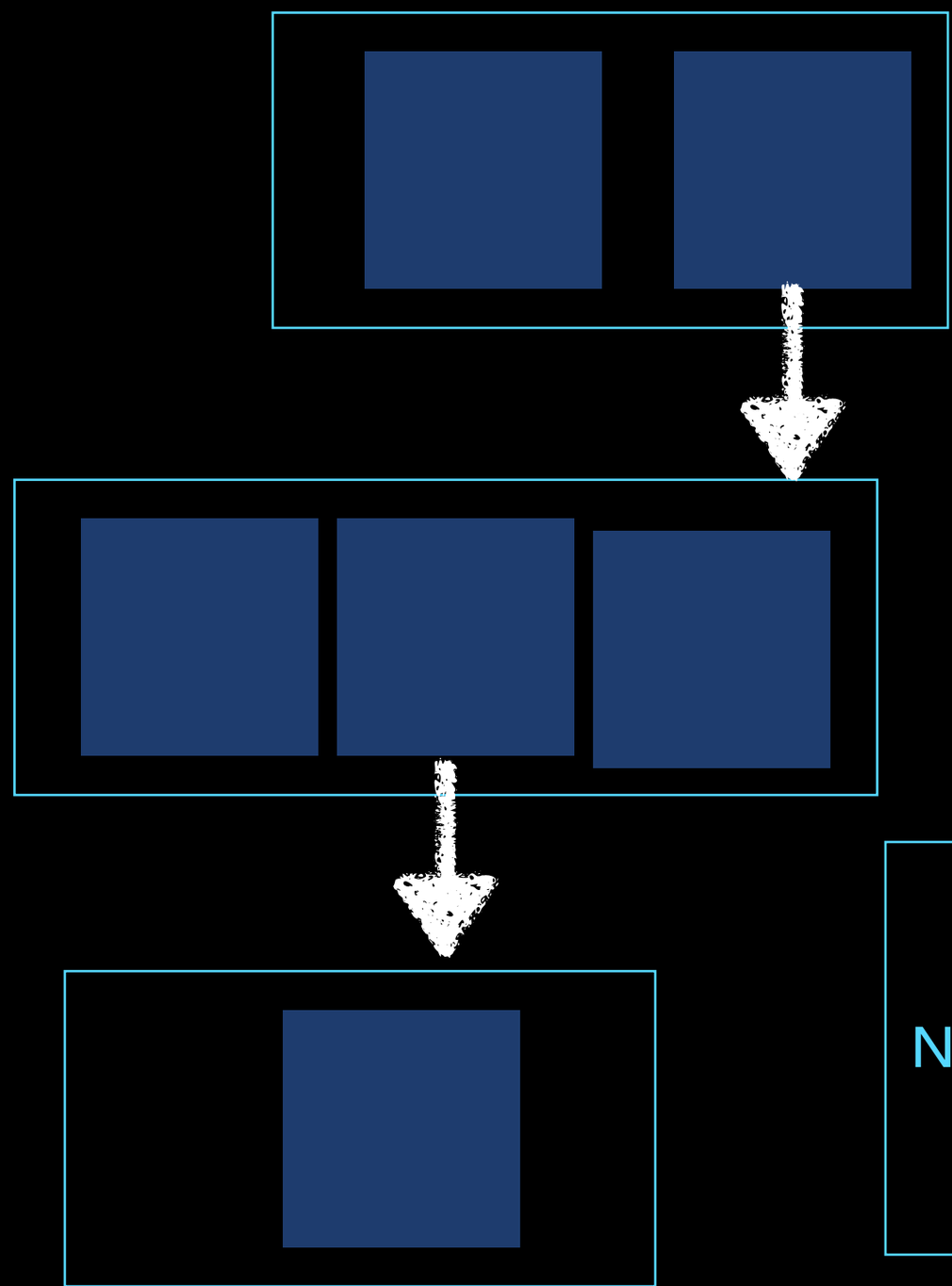
GET ME ALL OF USER FROM USERS COLLECTION THAT HAS WRITTEN REVIEW FOR RESTAURANT 2



RULES FOR QUERIES

- Queries need to be equality or greater than equal type commands
  - Example find all diners name = ‘ohill’
  - Find all reviews rating >= 4
  - Can’t query based on calculation. Rating rating /2

ALL QUERIES ARE SHALLOW



YOU WILL NOT GET
THE SUB COLLECTION

TIME IT TAKES
RUN A QUERY IS PROMOTIONAL
NUMBER OF RESULT YOU GET BACK
NOT THE NUMBER OF DOCS
YOU SEARCH THROUGH

TIME IT TAKES
RUN A QUERY IS PROMOTIONAL
NUMBER OF RESULT YOU GET BACK
NOT THE NUMBER OF DOCS
YOU SEARCH THROUGH

If you only 6 result back it doesn’t matter
If search 6 Million or 60 records
It will take the same amount of time

All fields and Maps in document are index

Easy to search
Through
Binary search

Rating > 3

RATING	DOC_ID
1	XXX
2	BBB
3	CCC
4	ADS

Even does this for maps

```
Name: "BurgerThyme!"
address: {
  addr_1: "123 Fake Street"
  city: "San Francisco"
  state: "CA"
  zip: "94505"
}
avg_rating: 4.76
Reviews: (Subcollection)
```

Firebase will create an index address_zip

Restaurant.name = "CAV"

NAME	DOC_ID
"ABE"	XXX
"CAV"	BBB
"CASTLE"	CCC
"OHILL"	ADS

- Finding diners will hill name harder there is no index

No Regular expression
Based search

NAME	DOC_ID
"ABE"	XXX
"CAV"	BBB
"CASTLE"	CCC
"OHILL"	ADS

NOT OR QUIRES

NO NILL
NOT EQUAL QUIRES

Requires linear sweep
Rule no slow quires

NAME	DOC_ID
"ABE"	XXX
"CAV"	BBB
"CASTLE"	CCC
"OHILL"	ADS

Possible to mix types, but will create Separate Indexes, don't mix types

Rating = 4

You will have to Query the Indies Separately

Rating == "4 OF 5"

RATING	DOC_ID
1	XXX
2	BBB
3	CCC
4	ADS

RATING	DOC_ID
"1 OF 5"	QQQ
"2 OF 5"	DDD
"3 OF 5"	YYY
"4 OF 5"	DSS

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = south

CAPACITY	DOC_ID
10	XXX
20	BBB
20	CCC
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC_ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

CAN QUERY MULTIPLE FIELDS AT ONCE

diner = 20 & location = North

CAPACITY	DOC ID
10	XXX
20	BBB
20	CCC
20	DDD
20	XXX
20	YYY
42	ADS

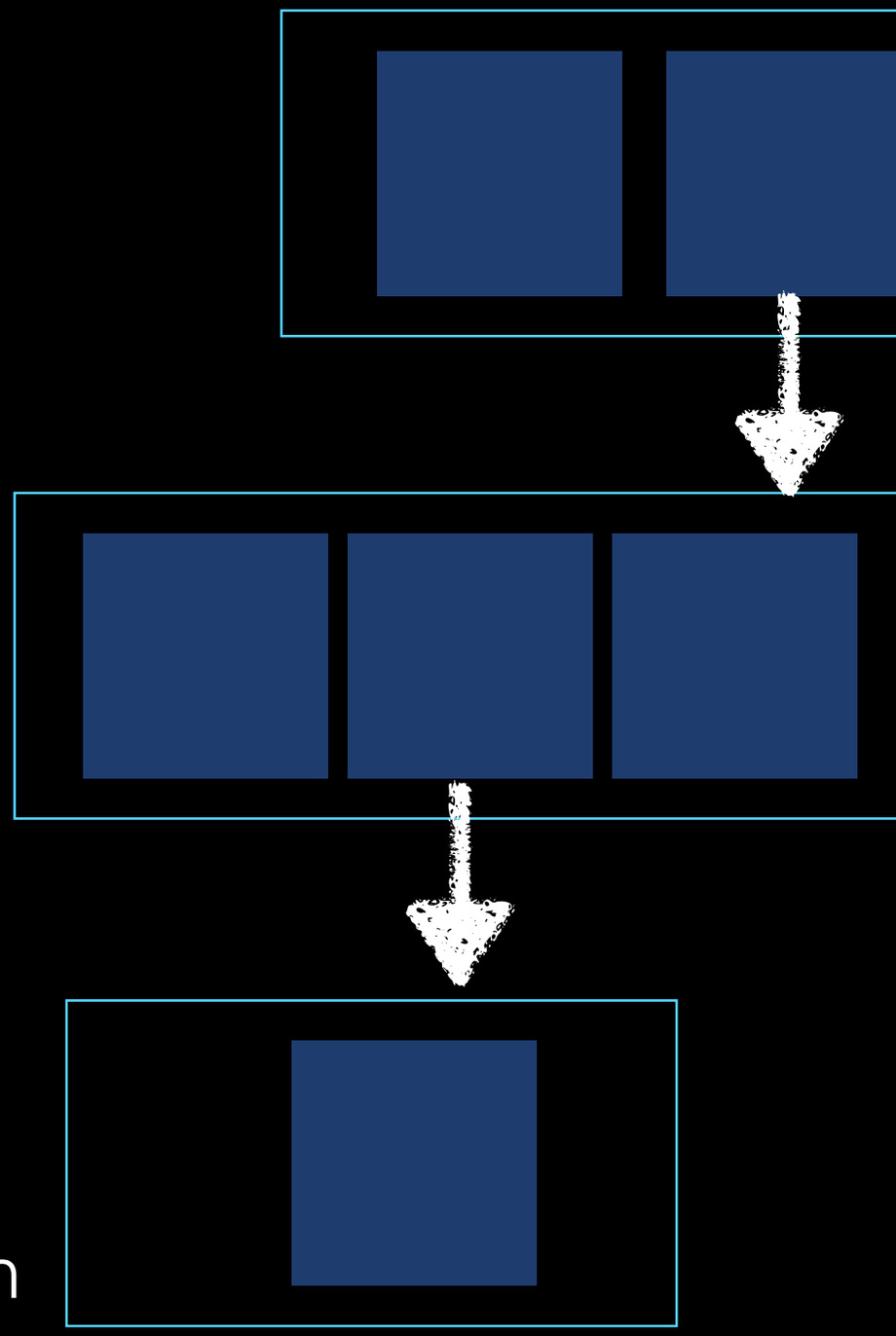
LOCATION	DOC_ID
NORTH	BBB
NORTH	XXX
CENTRAL	CCC
SOUTH	ADS

First sort by key and then doc id

Algorithm terminates

HOW TO STRUCTURE YOU DATA

- The Rules
  - Document have size limit 1MB
  - Can retrieve a partial document
  - Queries are shallow
  - Billed by number of read and writes you perform
  - Queries find documents in collection

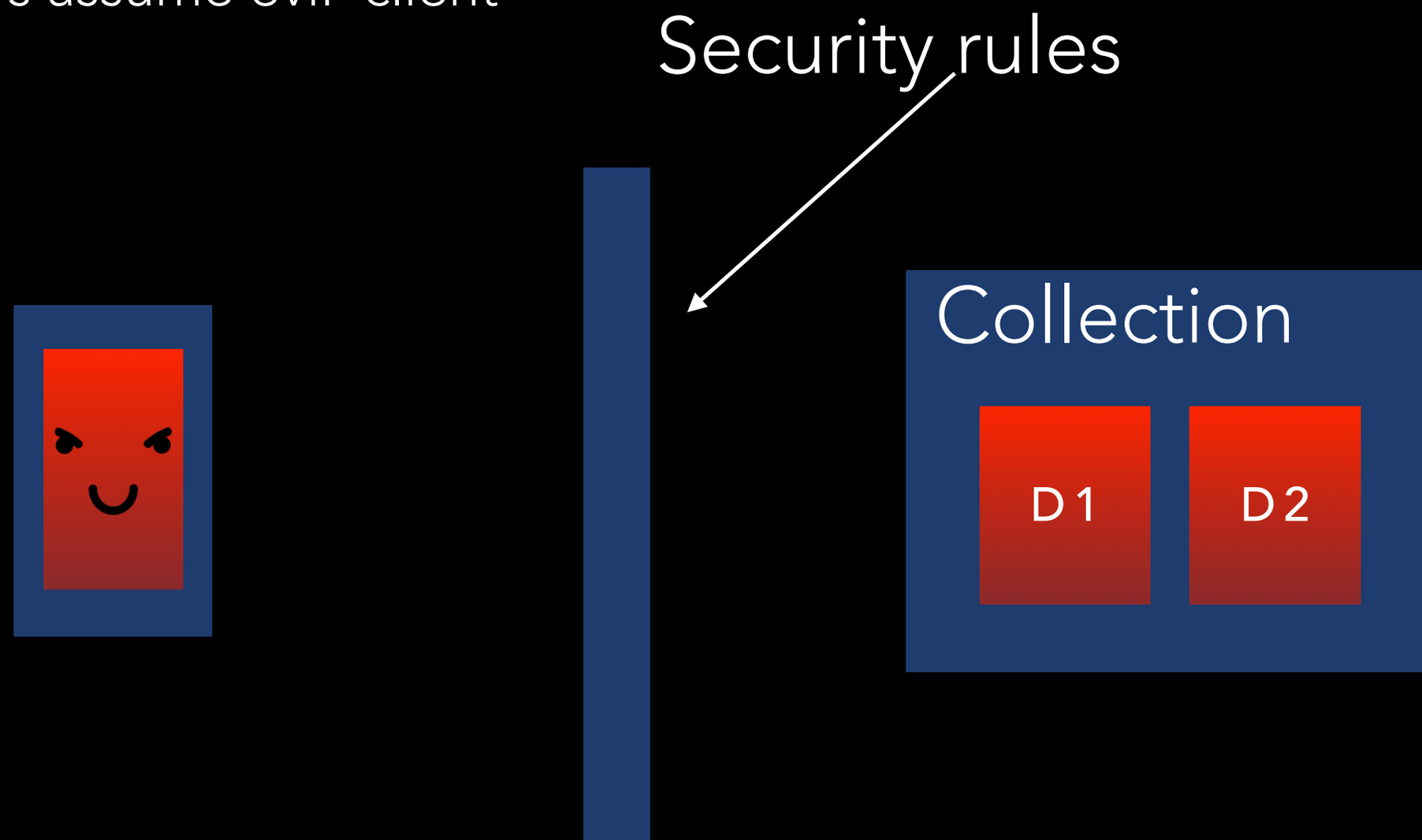


SECURITY RULES

- Can give every use complete access to the database otherwise the can do really bad stuff. Like edit or delete other users data.

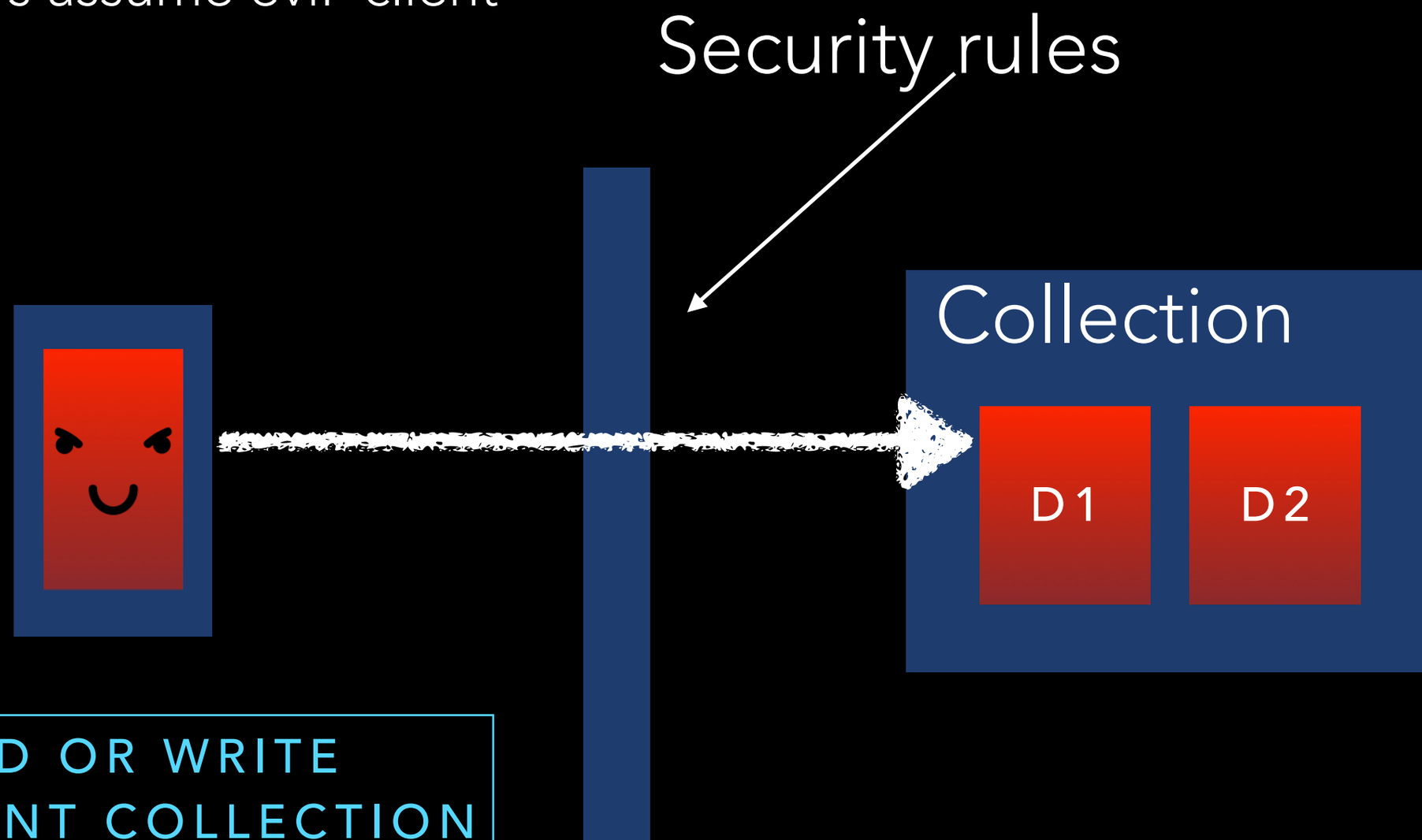
HOW DO WE DEAL WITH THIS? CLIENT LEVEL SECURITY

- Never trust your client application.
- Always assume evil client



HOW DO WE DEAL WITH THIS? CLIENT LEVEL SECURITY

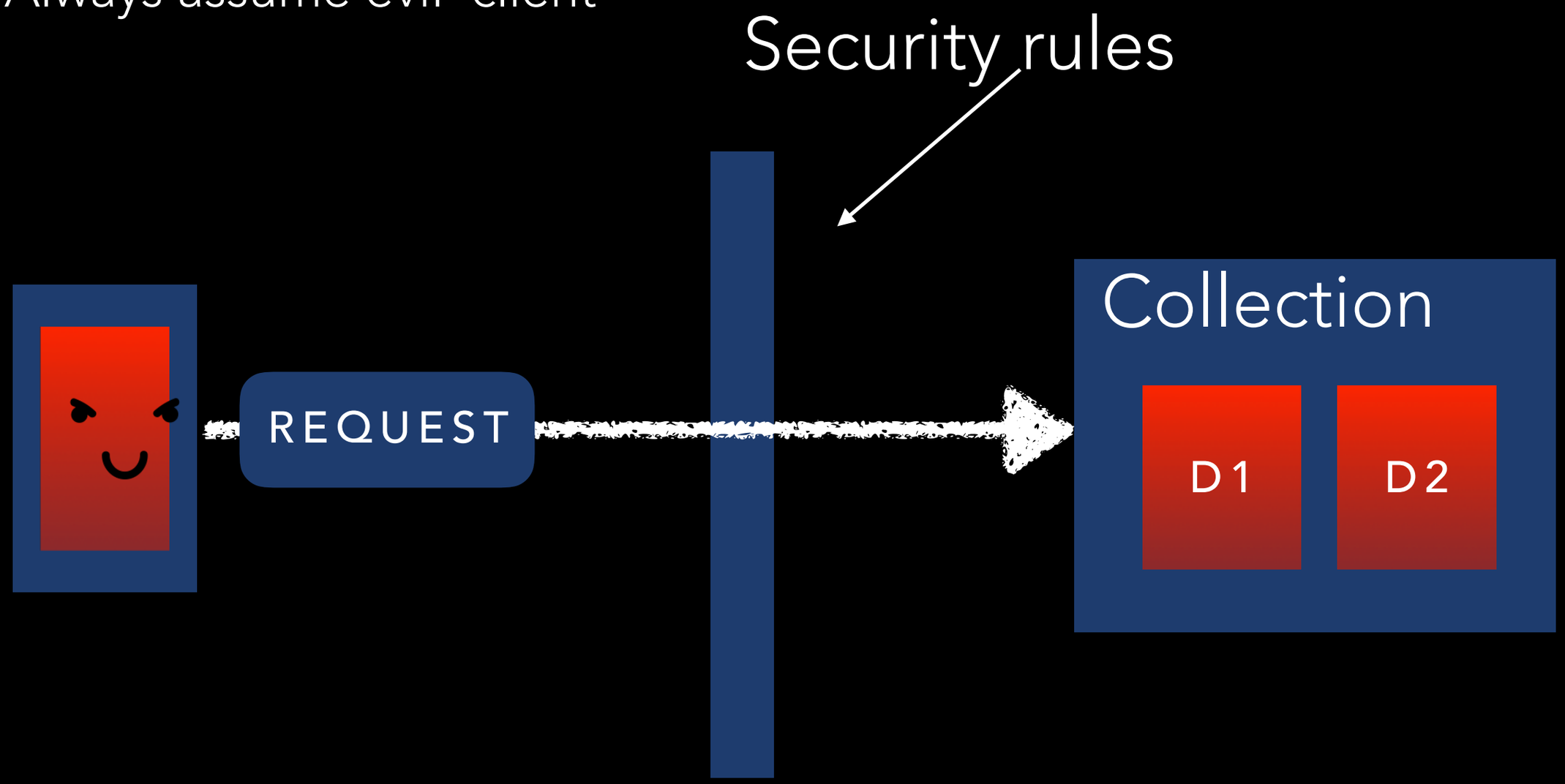
- Never trust your client application.
- Always assume evil client



READ OR WRITE
DOCUMENT COLLECTION

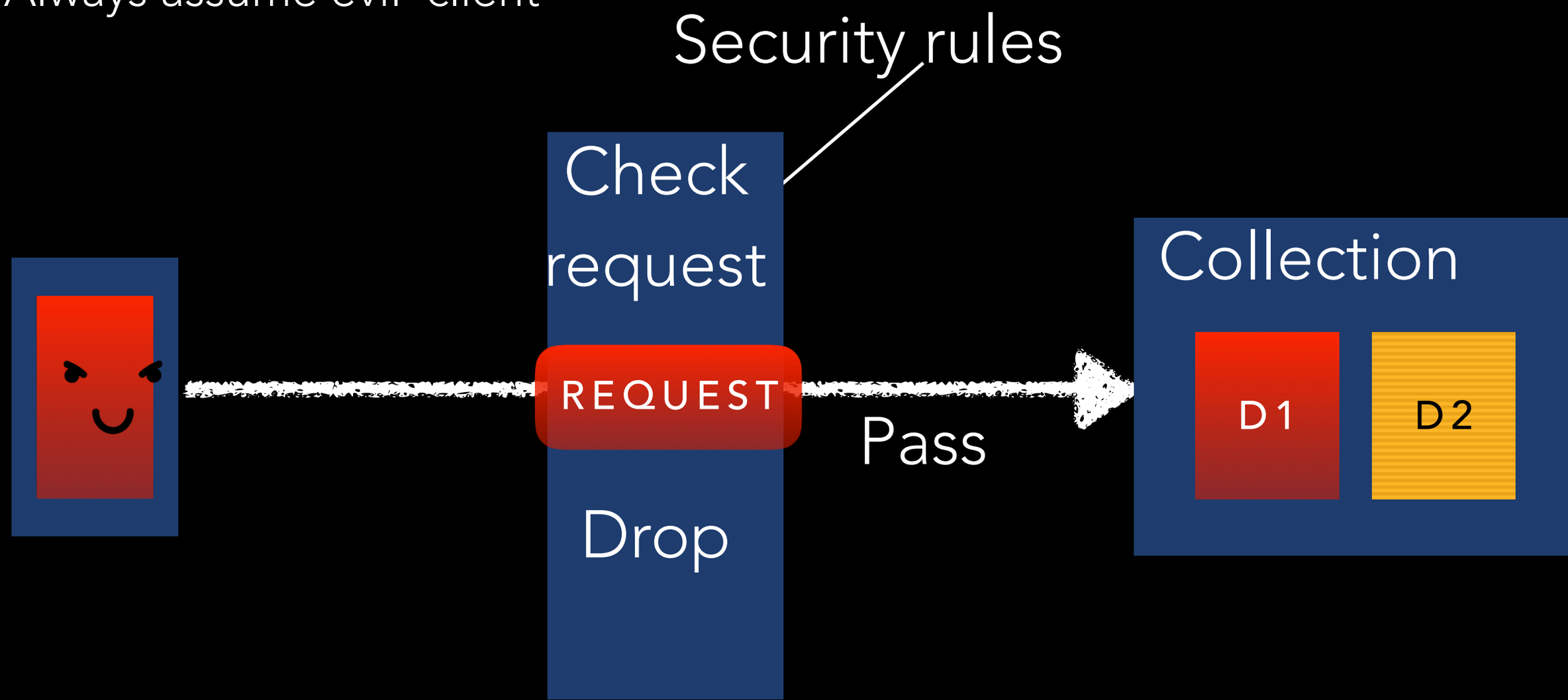
HOW DO WE DEAL WITH THIS? CLIENT LEVEL SECURITY

- Never trust your client application.
- Always assume evil client



HOW DO WE DEAL WITH THIS? CLIENT LEVEL SECURITY

- Never trust your client application.
- Always assume evil client



WHAT DOCUMENT ARE SECURING AND WHAT LOGIC ARE USING TO SECURE THEM

HOW DO WE DEAL WITH THIS? CLIENT LEVEL SECURITY

MATCH ANY DATABASE NAME

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      // Completely locked
      allow read, write: if false
    }
  }
}
```


```
databases/<database_name>/documents/restaurants/todd_tacos
```

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {

    }
    match /restaurants/{restaurantID}/reviews/{reviewID} {

    }
  }
}
```

Allowed to paths for rule Matches

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {
      match /reviews/{reviewID} {

      }
    }
  }
}
```

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {
      match /reviews/{reviewID} {

      }
      match /private-data/{privateDoc} {

      }
    }
  }
}
```

Parallel Paths

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {
      match /reviews/{reviewID} {

      }
      match /private-data/{privateDoc} {

      }
    }
  }
}
```

Rules at this top
Level does apply
For nested levels

Best practice to specify rules documents not Collections

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants {

    }
  } ^
}
```

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {

    }
  } ^
}
```

Add this wild card
To represent all
Documents

Two types of wild cards: Type 1 Single element wild card

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {

    }
  }
}
```

Single element wildcard

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {
      // restaurantID = The ID of the restaurant doc
      match /reviews/{reviewID} {
        // restaurantID = The ID of the restaurant doc
        // reviewID = The ID of the review doc
        allow write: if reviewID == "review_234"
      }
    }
  }
}
```

Match everything in rest of the path

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{restOfPath=**} {
      |
    }
  }
}
```

EVERY DOCUMENT IN THE USER COLLECTION AND ALL SUB COLLECTIONS

Value of the restOfPath contains eg:

/collection/document/collection

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{restOfPath=**} {
      allow read;
    }
    match /users/{userID}/privateData/{privateDoc} {
      // Doesn't do anything
      allow read: if false;
    }
  }
}
```

TOP RESULT TAKE PRESIDENCE

5 Conditions that you check

1. Get (Get document)
2. List (Get documents in collection

} READ

3. Create (Create a document)
4. Delete (Delete a document)
5. Update (Update content of a document)

} WRITE

FOR EACH ACTION YOU WILL RETURN A BOOLEAN TO INDICATE WHERE IT IS POSSIBLE TO EXECUTE THAT ACTION

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /publicDocs/{docID} {
      allow read: if true;
    }
  }
}
```

ALL DOCS IN THE PUBLIC COLLECTION CAN BE READ BY ANYONE

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /publicDocs/{docID} {
      allow read;
      allow write: if false;
    }
  }
}
```

ALL OTHER ACTIONS ARE FALSE BY DEFAULT

Normally you make decisions based on a condition

1. Based on the request data
2. Based on the target document
3. Based on other data in database

Normally you make decisions based on a condition

1. Based on the request data
2. Based on the target document
3. Based on other data in database

Request

THOUGH THE AUTH REQUEST IS COMING COMING FROM USER YOU CAN GENERALLY TRUST IT BECAUSE IT IS GENERATE BY A PREVIOUS AUTHENTICATION EXCHANGE

Normally you make decisions based on a condition

1. Based on the request data
2. Based on the target document
3. Based on other data in database

Request

request.auth !=null

request.auth.uid

request.auth.token.email

request.auth.token.email_verified

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /myCollection/{docID} {
      allow read: if request.auth != null;
    }
  } ^
}
```

ONLY READ DATA IF YOU ARE SIGNED IN

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /myCollection/{docID} {
      allow read: if request.auth.token.email.matches('.*google[.]com$') &&
        request.auth.token.email_verified;
    }
  }
}
```

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /myCollection/{docID} {
      allow read: if request.auth.uid == "albert_24"
    }
  }
}
```

ONLY ALBERT_24 CAN READS THESE DOCUMENT

What about writes?

Request

You can access any of the files in the data
As you would properties in a JSON object

request.resource.data.property

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /restaurants/{restaurantID} {
      match /reviews/{reviewID} {
        allow create: if request.resource.data.score is number &&
          request.resource.data.score >= 1 &&
          request.resource.data.score <= 5 &&
          request.resource.data.headline is string &&
          request.resource.data.headline.size() > 2 &&
          request.resource.data.headline.size() < 200 &&
          request.resource.data.review is string &&
          request.resource.data.review.size() > 20 &&
          request.resource.data.review.size() < 2000 &&
          request.resource.data.reviewerID == request.auth.uid;
      }
    }

  }
}
```

Important

User be able to submit review
On behalf of other users

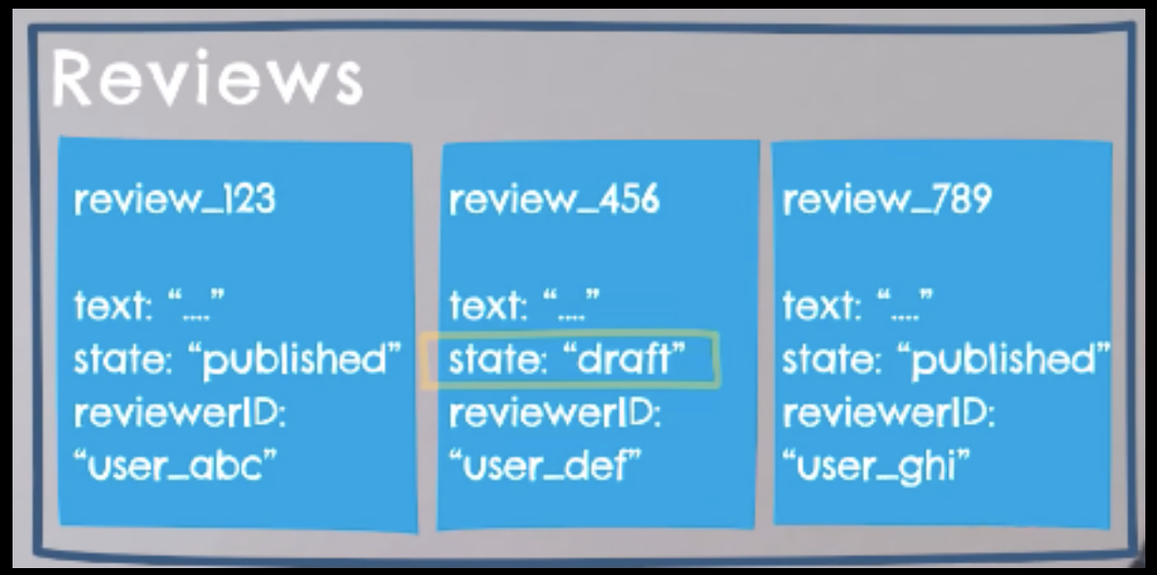
Request

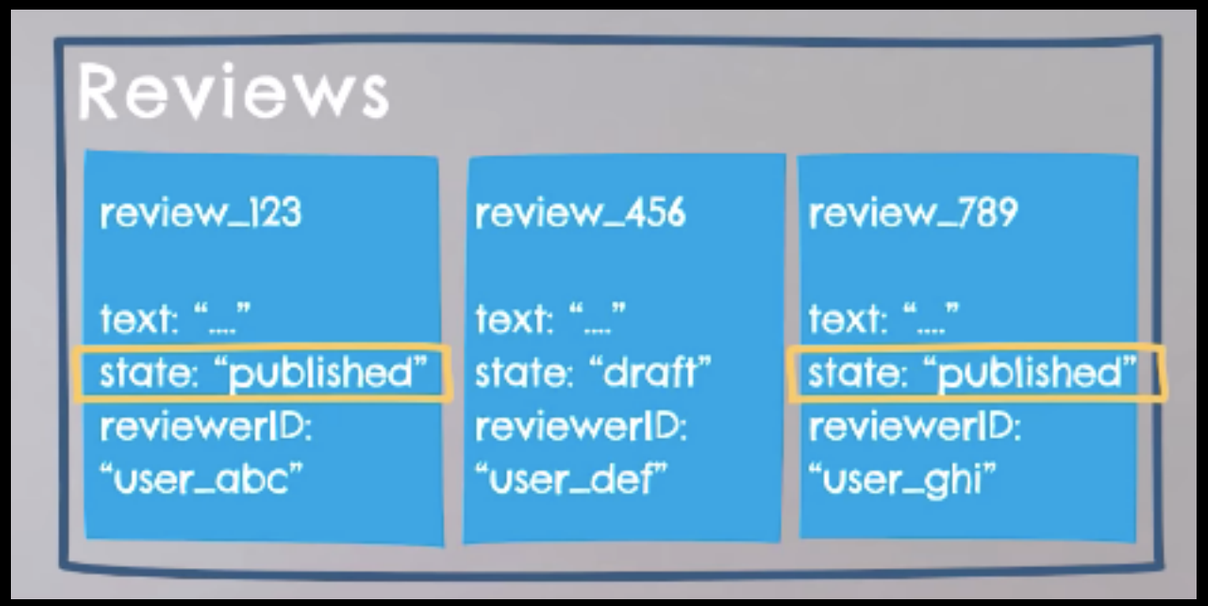
There also other resource object
That represents the data being written
Or read that is already in the database

RESOURCE NOT THE SAME AS REQUEST.RESOURCE

```
allow update: if request.resource.data.score is number &&
  request.resource.data.score >= 1 &&
  request.resource.data.score <= 5 &&
  request.resource.data.headline is string &&
  request.resource.data.headline.size() > 2 &&
  request.resource.data.headline.size() < 200 &&
  request.resource.data.review is string &&
  request.resource.data.review.size() > 20 &&
  request.resource.data.review.size() < 2000 &&
  request.resource.data.reviewerID == request.auth.uid &&
  resource.data.reviewerID == request.auth.uid &&
  // Can't change the score.
  request.resource.data.score == resource.data.score;
}
```

YOU CAN ONLY REVIEWS THAT YOU HAVE WRITTEN BECAUSE YOUR USERID NEEDS TO MATCH THE RESOURCE ID





Make sure that User can't view Draft reviews

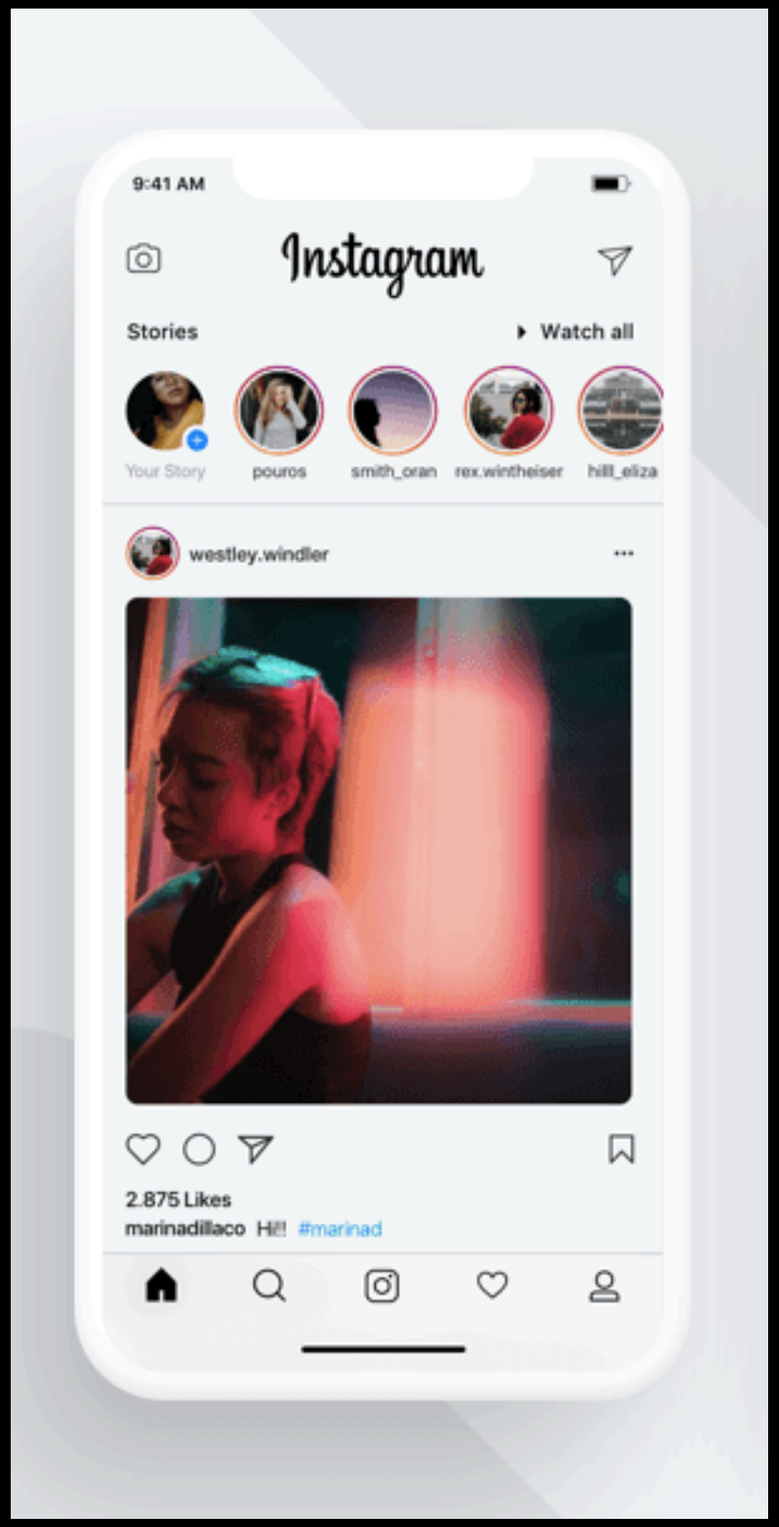
Can’t use security rules to filter for you

```
  allow read: if resource.data.state == "published" ||
    resource.data.reviewerID == request.auth.uid;
}
```

The query get all reviews would fail because the security rules does not evaluate each document. (Too much time)

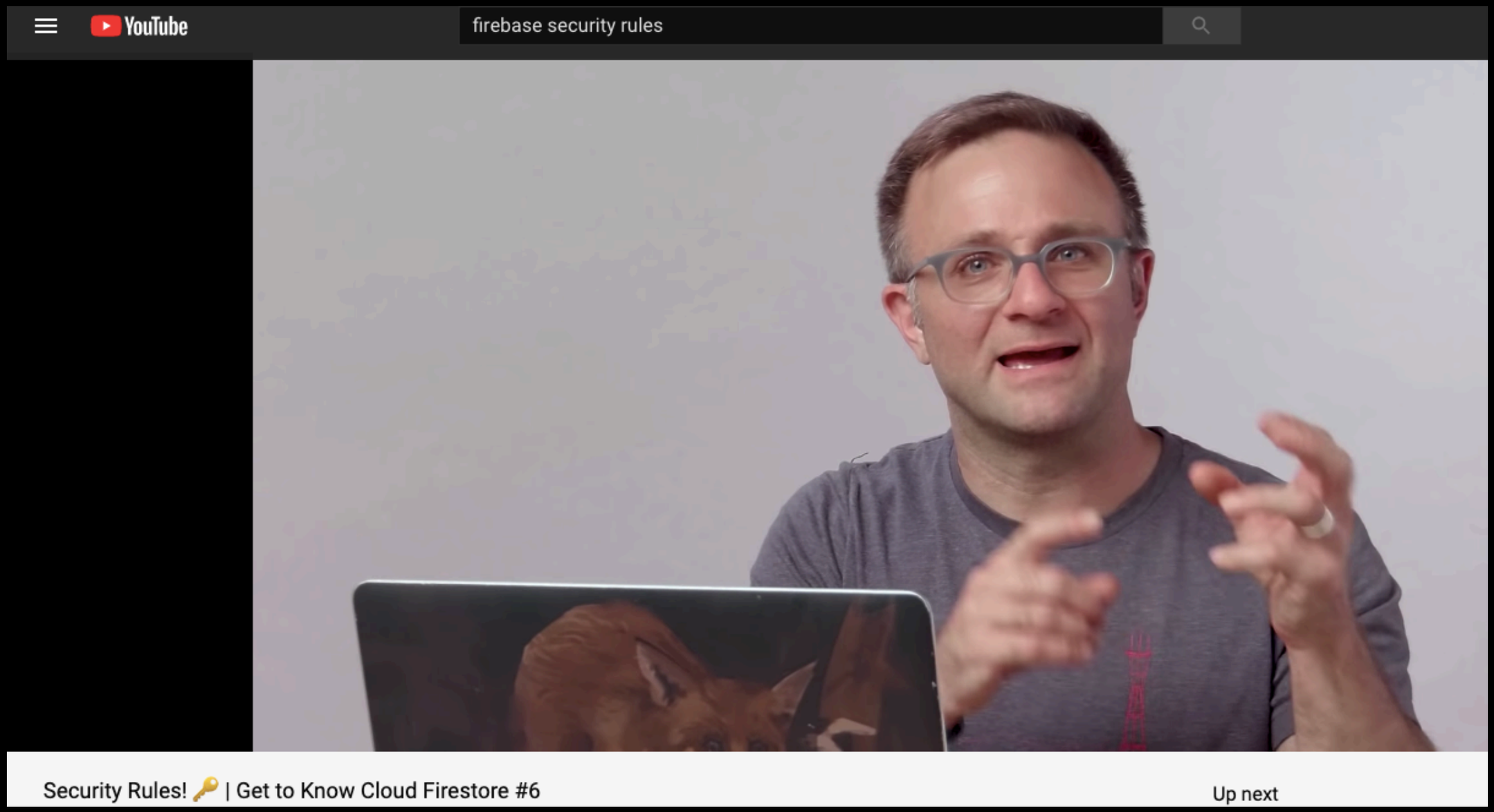
Look at the query and try determine if would be approved regardless of the content in the database

HOW TO STRUCTURE DATA



What documents
And collections would we need
To build instagram

This about security rules



https://www.youtube.com/watch?v=eW5MdE3ZcAw&t=247s