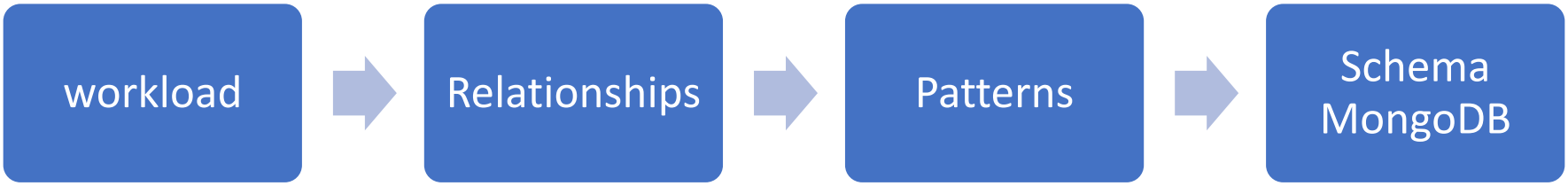
MongoDB

Database Schema Design

Lecture (3)

Schema Design



Identify and Quantify Entities

- **Book Store Application Entities**
  - eBooks {SKU, title, author, publisher, language, price, summary, rating, release date, **pages**}
  - Audibook {.., duration, narrators}
  - Printed book {.., pages, stockLevel, deliveryTime}
  - Authors {authorId, name, birth, biography, socials}
  - Publishers {publisherId, name, founded, description, headquarters, socials}
  - Users {userId, name, email, phone, address, membersince}
  - Reviews {productSKU, userId, date, starts, reviewTitle, reviewBody}

Identify Reads and Writes!

- Type of operations
- Frequency of Operations

Entities	operation	Information needed	type	Frequency
Books	Fetch book details	Book details + rating	Read	1000/sec
Authors, books	Fetch an author and their books	Book title + author detail	Read	50/sec
Books	Add/update	Book details + stock level	Write	10/hour
Print books	Sell copy of printed book	Stock level	Write	5/sec
Reviews	Fetch top reviews of a book	Reviews + rating	Read	200/sec
Reviews	Add review	Review + book rating	Write	
Users	Fetch user details	User details	Read	

Example: Bakery Application

The bakery application is a comprehensive digital solution designed to streamline the management of a bakery operations, from inventory to customer engagement. It supports a variety of entities

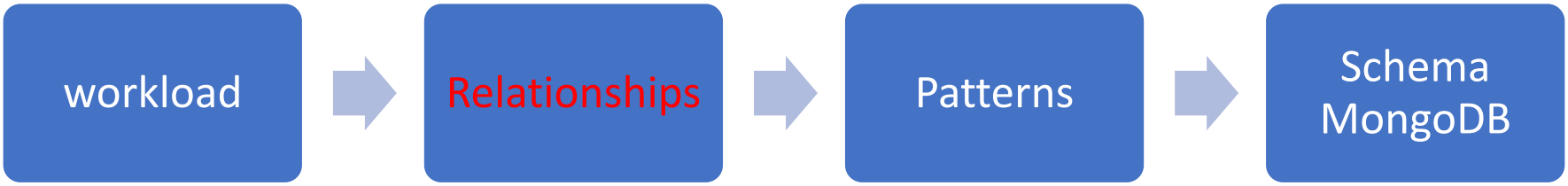
Bakery Store Application Entities

- Product {productid, name, category, price, ingredients, weight, shelfLife, availability}
- Bakery {bakeryId, name, location, operatingHours, contactInfo, owner, services}
- Customers {custoomerId, name,email,phone, loyaltypoints, allergies, preferreditems}
- orders {orderId, customerId, orderDate, deliveryDate, productsOrdered, totalAmount, orderStatus}
- Suppliers {supplierId, name, contactinfo, productsupplied, deliveryschedule, paymentTerms}
- Employees {employeeId, name, role, contactinfo, hoursWorked, salary}
- Feedback {productid, customerId, date, rate, review}

Workload

Entities	operation	Information needed	type	Frequency
Products	Fetch product details	Product details + availability	Read	500/sec
Orders	Fetch recent orders	Order details + customer info	Read	20/min
Customers	Fetch customer details	Customer loyalty points+ preferences	Read	100/hr
Products	Add/update product	Product details + ingredients	Write	30/hr
Bakery inventory	Update stock levels	Product stock levels	write	10/day
Employees	Update employee hours	Employee work schedule	Write	5/hr
Suppliers	Fetch supplier details	Supplier products + delivery schedule	Read	3/day
Orders	Add new order	Order details + customer info	Write	50/day
Customer feedback	Add customer review	Review + product rating	Write	10/day
Inventory alert	Fetch product with low stock	Product details + stock level	Read	2/day

Schema Design



Relationships

- One-to-one
- One-to-many
- Many-to-many
- Embedded vs referencing
- Modeling details

Identify relationships

- Publishers {publisherId, name, founded, description, **headquarters**, socials}
- Headquarters {street, city, state, postal, country}

One-to-one

Identify relationships

- Printed books
- Reviews

One-to-many

Identify relationships

- Printed books
- authors

Many-to-many

Embedded vs referencing

- **Reference**: separate document linked using a key
- **Embed**: both entities inside each other
- Selection Criteria!

Rule of thumb: Data that is accessed together stored together!

Guidelines

- Eg.: Printed books vs authors (many-to-many)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	yes	no
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	yes	no
Query Atomicity	Does the application query the pieces of information together?	yes	no
Update Complexity	Are the pieces of information updated together?	yes	no
Archival	Should the pieces of information be archived at the same time?	yes	no
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	no	yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	no	yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	no	yes
Document Growth	Would the embedded piece grow without bound?	no	yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	no	yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	no	yes

Modeling 1-to-1

- Publisher – headquarters

Embedded option

```
Publisher {
        ”_id”:      ,
        “name”:  ,
        “headquarters”: {
                “street”:          ,
                “city”:            ,
                “state”:           ,
                ”postalcode”:      ,
                “country”:         ,
                }
        }
```

Reference option

```
Publisher {
        ”_id”:      ,
        “name”:  ,
        “hq_id”: “B11”      }
```

```
Headquarters {
                “_id”: “B11”       ,
                “street”:          ,
                “city”:            ,
                “state”:           ,
                ”postalcode”:      ,
                “country”:         ,
                }
```

Modeling 1-to-1

- Publisher – headquarters

Embedded option (11 points)

```
Publisher {
        ”_id”:      ,
        “name”:  ,
        “headquarters”: {
                “street”:          ,
                “city”:            ,
                “state”:           ,
                ”postalcode”:      ,
                “country”:         ,
                }
        }
```

Reference option

```
Publisher {
        ”_id”:      ,
        “name”:  ,
        “hq_id”: “B11”      }
```

```
Headquarters {
                “_id”: “B11”       ,
                “street”:          ,
                “city”:            ,
                “state”:           ,
                ”postalcode”:      ,
                “country”:         ,
                publisherId:       ,
                }
```

Modeling 1-to-N

- Printed books – reviews embedded option

```
book {
        ”_id”:      ,
        “title”:    ,
        “author”: ,
        ”reviews”: [
        {
          ‘user_id’:          ,
          ‘title”:            ,
          ‘review’:           ,
          ‘ratinig’:          ,
        }, ..
}
```

Reference option

```
book {
        ”_id”:      ,
        “title”:    ,
        “author”: ,
        “reviews”: [“r_01”, “r_02”] //array option
}
```

```
review {
        ‘_id’:”r_01”        ,
        ‘user_id’:          ,
        ‘title”:            ,
        ‘review’:           ,
        ‘ratinig’:          ,
        ‘book_id’:          , //reference parent
                }
```

Modeling M-to-N

- Printed books – authors

embedded option (9 points)

```
book {
        ”_id”:      ,
        “title”:    ,
        ”authors”: [
        {
          ‘author_id’:        ,
          ‘name”:             ,
        },
        {‘author_id’:         ,
          ‘name”:             ,
        },..]
}
```

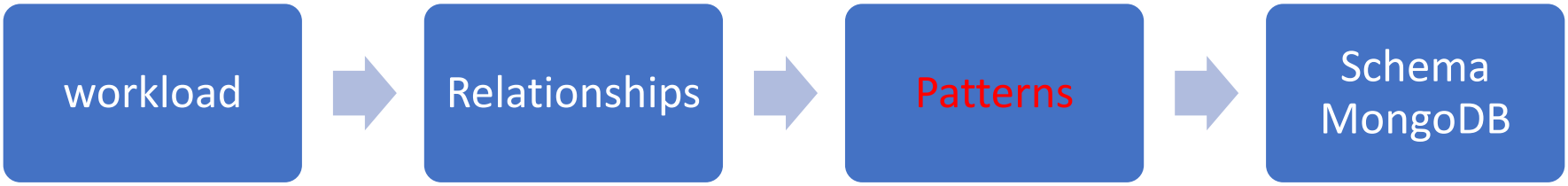
```
book {
        ”_id”:      ,
        “title”:    ,
        ”authors”: [
        {
          ‘author_01’:{
                ‘name”:   ,
                },
          ‘author_02’:{
                ‘name”:   ,
                },..]
}
```

Reference option

```
book {
        ”_id”:      ,
        “title”:    ,
        }
```

```
author {
        ”author_id”:        ,
        “name”:             ,
        ”books”: [
          ‘book_01’,
          ‘book_02’,
                …}
```

Schema Design



Inheritance pattern

- Different forms of similar entities
- Shared fields
- Unique fields
- More similarities than differences
  - Keep documents in the same collection

- Printed books vs audiobooks vs ebooks