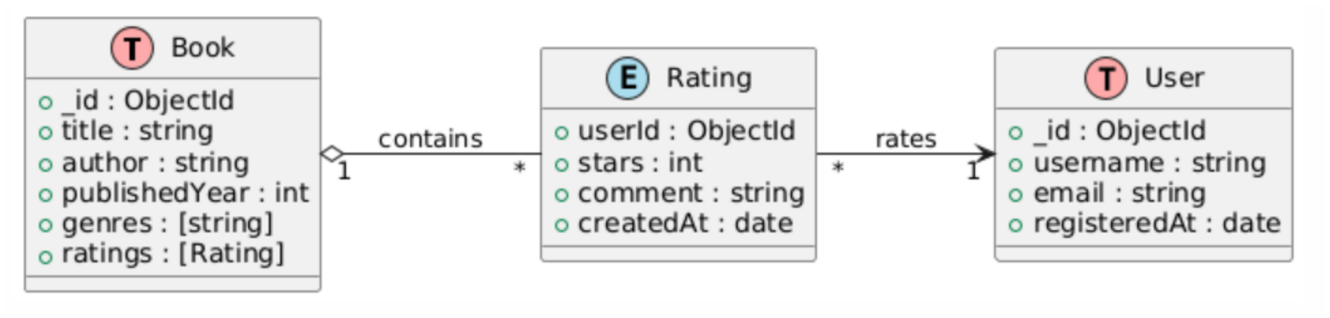
B1F Praktischer Nachweis

ERD Ausgangslage



- Bücher haben Titel, Autor, Erscheinungsjahr, Genres.
- Nutzer können Bücher bewerten (1–5 Sterne) und einen Kommentar hinterlassen.
- Jede Bewertung gehört zu einem Buch und einem Nutzer.

Struktur des Datenmodells in MongoDB

User

```
{
  "_id": ObjectId,
  "username": "booklover42",
  "email": "reader@example.com",
  "registeredAt": ISODate
}
```

Book

```
{
  "_id": ObjectId,
  "title": "Clean Code",
  "author": "Robert C. Martin",
  "published_year": 2008,
  "genres": ["Programming", "Software Engineering"]
}
```

Rating

```
{
  "_id": ObjectId,
  "book_id": ObjectId, // Referenz zum Buch
  "user_id": ObjectId, // Referenz zum Nutzer
  "user_id": ObjectId, // Referenz zum Nutzer
  "stars": 5,
  "comment": "Fragwürdige Empfehlungen!",
  "created_at": ISODate
}
```

Mit Container von oben Verbinden und in `library` Datenbank wechseln:

```
docker exec -it mongodb mongosh
use library
```

Benutzer anlegen

```
# Benutzer anlegen
const userId = ObjectId();
db.users.insertOne({
  _id: userId,
  username: "reader123",
  email: "reader@example.com",
  registered_at: new Date()
});
```

Buch anlegen

```
const bookId = ObjectId();
db.books.insertOne({
  _id: bookId,
  title: "Clean Code",
  author: "Robert C. Martin",
  published_year: 2008,
  genres: ["Programming", "Software Engineering"]
});
```

Bewertung erstellen

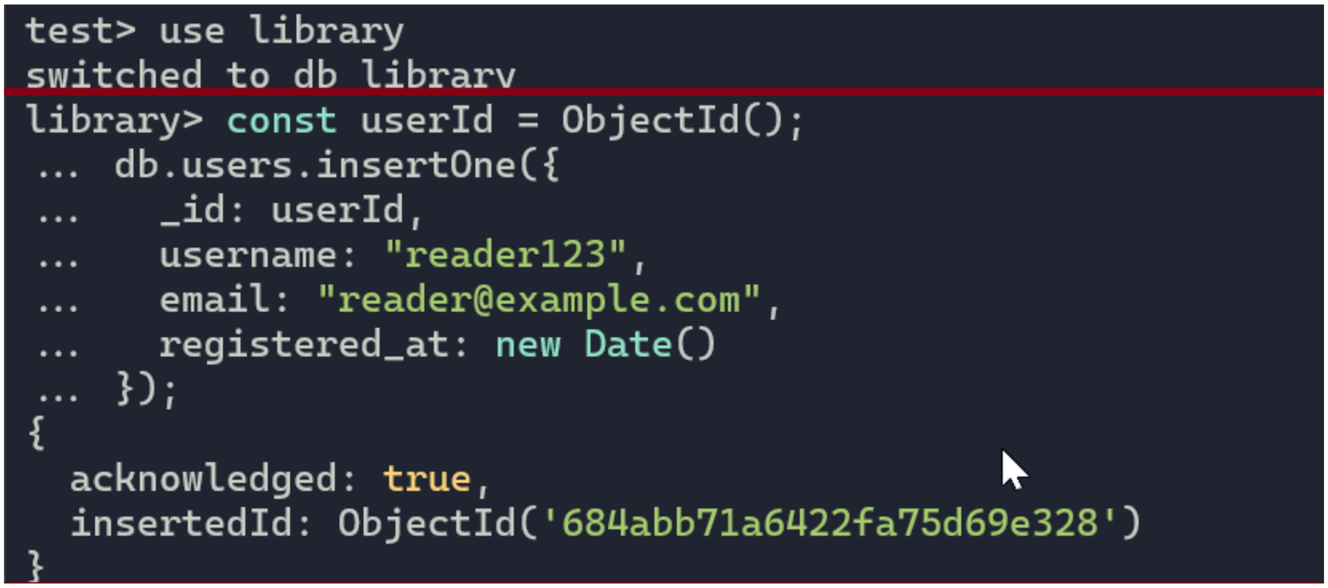
```
db.ratings.insertOne({
  book_id: bookId,
  user_id: userId,
  stars: 5,
  comment: "Fragwürdige Empfehlungen!",
  created_at: new Date()
});
```

Abfragebeispiel mit Aggregation

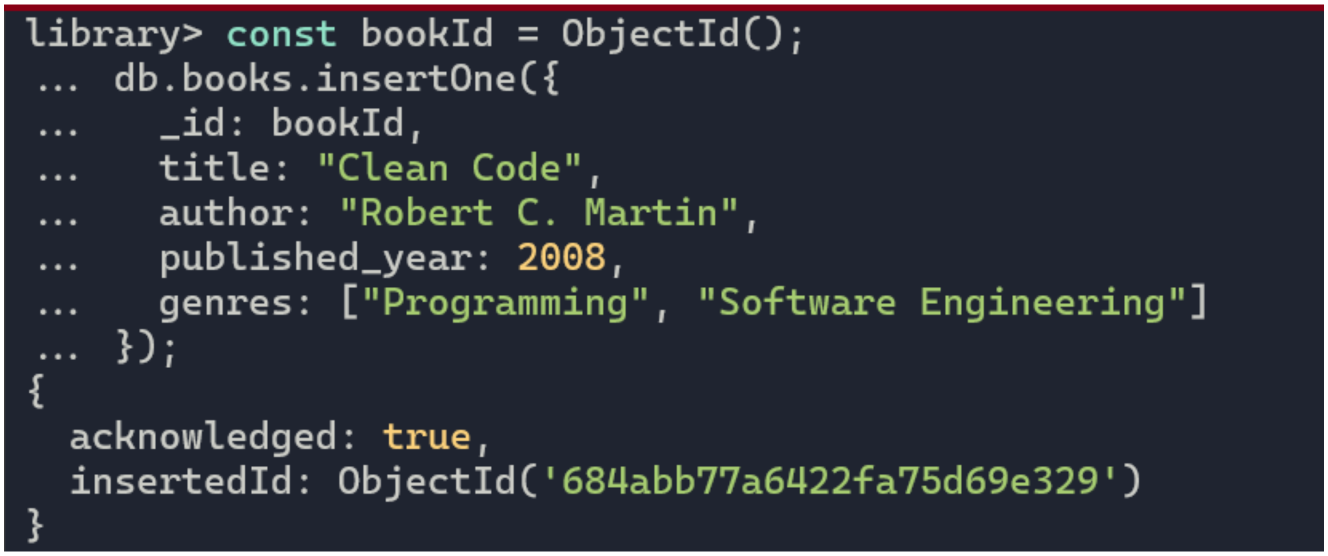
```
db.ratings.aggregate([
db.ratings.aggregate([
  {
    $lookup: {
      from: "books",
      localField: "book_id",
      foreignField: "_id",
      as: "book"
    }
  },
  { $unwind: "$book" },
  {
    $lookup: {
      from: "users",
      localField: "user_id",
      foreignField: "_id",
      as: "user"
    }
  },
  { $unwind: "$user" },
  {
    $project: {
      book_title: "$book.title",
      reviewer: "$user.username",
      stars: 1,
      comment: 1,
      created_at: 1
    }
  }
]);
```

Screenshots

Insert Book and User

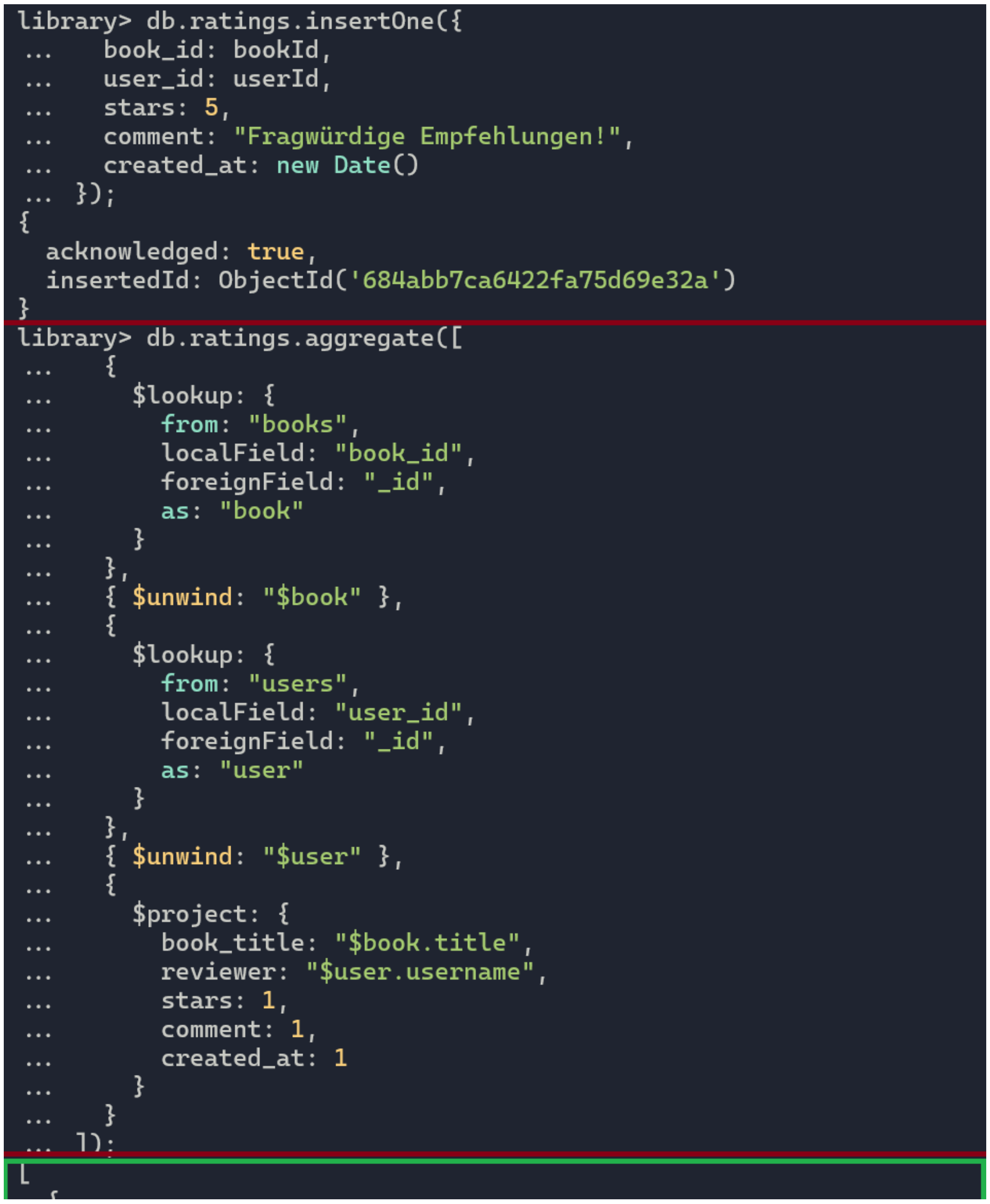


```
test> use library
switched to db library
library> const userId = ObjectId();
... db.users.insertOne({
...   _id: userId,
...   username: "reader123",
...   email: "reader@example.com",
...   registered_at: new Date()
... });
{
  acknowledged: true,
  insertedId: ObjectId('684abb71a6422fa75d69e328')
}
```

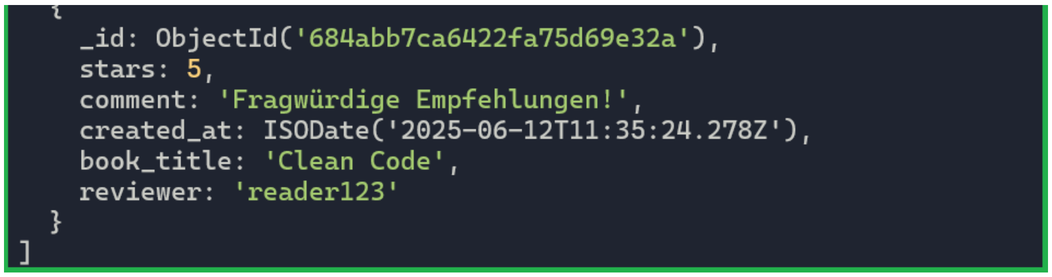


```
library> const bookId = ObjectId();
... db.books.insertOne({
...   _id: bookId,
...   title: "Clean Code",
...   author: "Robert C. Martin",
...   published_year: 2008,
...   genres: ["Programming", "Software Engineering"]
... });
{
  acknowledged: true,
  insertedId: ObjectId('684abb77a6422fa75d69e329')
}
```

Insert rating and show aggregation



```
library> db.ratings.insertOne({
...   book_id: bookId,
...   user_id: userId,
...   stars: 5,
...   comment: "Fragwürdige Empfehlungen!",
...   created_at: new Date()
... });
{
  acknowledged: true,
  insertedId: ObjectId('684abb7ca6422fa75d69e32a')
}
library> db.ratings.aggregate([
...   {
...     $lookup: {
...       from: "books",
...       localField: "book_id",
...       foreignField: "_id",
...       as: "book"
...     }
...   },
...   { $unwind: "$book" },
...   {
...     $lookup: {
...       from: "users",
...       localField: "user_id",
...       foreignField: "_id",
...       as: "user"
...     }
...   },
...   { $unwind: "$user" },
...   {
...     $project: {
...       book_title: "$book.title",
...       reviewer: "$user.username",
...       stars: 1,
...       comment: 1,
...       created_at: 1
...     }
...   }
... ]);
[
```



```
  {
    _id: ObjectId('684abb7ca6422fa75d69e32a'),
    stars: 5,
    comment: 'Fragwürdige Empfehlungen!',
    created_at: ISODate('2025-06-12T11:35:24.278Z'),
    book_title: 'Clean Code',
    reviewer: 'reader123'
  }
]
```