请针对以下需求设计一个文档数据库的模式：  
一个关于电影、制作人员和演员的网站（类似一个简易的IMDB网站）。用户可以浏览每一部电影的简介和相关信息（出品年份、电影类型、时长、评级），以及其导演、编剧和演员的列表。用户还可以浏览每一位导演、编剧或演员的信息（姓名、性别、年龄、简介），以及他们参与过哪些电影作品。导演、编剧或演员只是职位，一个人可以身兼数职。用户登录后还可以针对每一部电影、每一位导演、编剧或演员进行评价和打分，供别人参考。

设计一个文档数据库模式，需要考虑如何有效地组织和存储数据以便能够满足查询需求，同时还需要支持灵活的数据更新。以下是一个简易的IMDB风格网站的文档数据库模式设计：

**电影文档结构（Movies Collection）：**

{

"\_id": "movie\_id",

"title": "电影标题",

"description": "电影简介",

"release\_year": "出品年份",

"genre": ["电影类型"],

"duration": "时长",

"rating": "评级",

"crew": {

"directors": [

{"person\_id": "director\_id", "name": "导演姓名"}

],

"writers": [

{"person\_id": "writer\_id", "name": "编剧姓名"}

],

"actors": [

{

"person\_id": "actor\_id",

"name": "演员姓名",

"role": "角色名"

}

]

},

"reviews": [

{

"user\_id": "user\_id",

"rating": 5,

"comment": "评论",

"date": "评论日期"

}

]

}

**人员文档结构（Persons Collection）：**

{

"\_id": "person\_id",

"name": "姓名",

"gender": "性别",

"age": "年龄",

"bio": "简介",

"filmography": [

{

"movie\_id": "movie\_id",

"title": "电影标题",

"role": ["director", "writer", "actor"],

"character": "角色名（如果适用）"

}

]

}

**用户文档结构（Users Collection）：**

{

"\_id": "user\_id",

"username": "用户名",

"password": "加密密码",

"email": "电子邮箱",

"reviews": [

{

"movie\_id": "movie\_id",

"rating": 5,

"comment": "评论",

"date": "评论日期"

},

{

"person\_id": "person\_id",

"rating": 5,

"comment": "评论",

"date": "评论日期"

}

]

}

在这个设计中：

电影文档包含了电影的基本信息以及一个指向制作人员的引用列表，每个人员有一个对应的ID，这样可以避免在电影文档中重复存储人员的详细信息。

人员文档包含了个人信息和参与过的电影作品的列表，通过电影ID进行关联。这允许用户查看某个人员的所有作品。

用户文档包含了用户信息和他们对电影及人员的评价。每个评价记录都关联到一个电影ID或人员ID。

这种模式设计允许灵活地添加新的电影、人员或用户评价，同时保持数据的规范性。要注意的是，实际的设计可能会更复杂，需要考虑性能优化、索引策略以及如何处理大量数据的更新和查询。在实际应用中，还可能需要考虑到数据的安全性、备份和恢复策略。

请在上一次作业中关于电影网站的数据苦设计基础上进行追加设计，实现以下功能：

每一位演员在其参演的电影中都扮演一定的角色。用户在浏览电影时，除了能看到演员信息，还能看到每位演员扮演了什么角色。用户在浏览演员时，除了能看到他（或她）参演的电影，还能看到他（或她）在每一部电影中扮演的角色。

注意：一位演员可以在一部电影中扮演多个角色。同一个角色也可能由多名演员扮演，比如，年少时由一位演员扮演，年老时由另一位演员扮演。

为了实现这些额外的功能，我们需要在现有的电影和人员文档结构中进一步添加角色信息。在此基础上，我将提供调整后的文档结构以支持这些新的需求。

**更新后的电影文档结构（Movies Collection）：**

{

"\_id": "movie\_id",

"title": "电影标题",

"description": "电影简介",

// ... 其他电影信息

"crew": {

"directors": [

{"person\_id": "director\_id", "name": "导演姓名"}

// ... 其他导演信息

],

"writers": [

{"person\_id": "writer\_id", "name": "编剧姓名"}

// ... 其他编剧信息

],

// 更新演员列表，包括角色信息

"actors": [

{

"person\_id": "actor\_id",

"name": "演员姓名",

"roles": [

{

"role\_name": "角色名",

"character\_age": "角色年龄段", // 如年少、年老

// ... 其他角色相关信息

}

// ... 其他角色信息

]

}

// ... 其他演员信息

]

},

// ... 评论信息

}

更新后的人员文档结构（Persons Collection）：

{

"\_id": "person\_id",

"name": "姓名",

// ... 其他个人信息

// 更新参演电影列表，包括角色信息

"filmography": [

{

"movie\_id": "movie\_id",

"title": "电影标题",

"roles": [

{

"role\_name": "角色名",

"character\_age": "角色年龄段",

// ... 其他角色相关信息

}

// ... 其他角色信息

]

}

// ... 其他电影信息

]

}

在这个更新后的设计中，我们为演员在电影中的角色创建了一个新的数组 roles，它包含了角色名和角色年龄段等信息。这允许我们存储一位演员在同一部电影中扮演多个角色的情况，也支持不同的演员扮演同一个角色的不同年龄段。这样，用户在浏览电影或演员时，都能看到详细的角色信息。

请注意，数据库设计应该是动态的，并随着应用程序需求的变化而变化。随着时间的推移和用户需求的发展，可能需要进一步的优化和调整。在实际的数据库实施过程中，还需关注性能、索引优化、查询优化以及数据完整性的问题。