北京理工大學

数据库设计与开发

作业三 数据库设计与实现

学　　院：	计算机学院
专　　业：	软件工程
班　　级：	08012301 班
学生姓名：	蒋浩天
学　　号：	1120231337
指导教师：	

2025 年 5 月 20 日

目 录

第 1 章 原创性说明

本作业系本人自主独立完成，特此申明!

第 2 章 作业要求

2.1 功能需求

在 OpenGauss 数据库中, 设计实现 “电脑诊所管理系统”, 要求能实现以下功能:

- 普通用户
  - 注册登录
  - 预约电脑维修
  - 查看预约状态
  - 取消预约
  - 查看公告
- 工作人员
  - 注册登录
  - 处理预约工单
- 管理员
  - 注册登录
  - 管理工作人员
  - 查看工单
  - 发布公告
  - 管理场地信息
  - 管理开放时间信息

2.2 设计说明

根据需求分析, 设计以下数据实体.

1. 普通用户

- 注册登录功能。
- 可选择场地和时间，填写个人信息后，发起预约工单
- 可查看自己的预约状态。
- 可取消预约。
- 可查看公告信息。

2. 工作人员
   - 类似普通用户。
   - 可在系统中修改工单信息（如完成、处理中）。
   - 可根据工单信息决定接受或拒绝预约
3. 管理员
   - 可管理工作人员名单与具体权限
   - 可在系统中修改场地信息和开放时间信息
   - 可管理公告信息
4. 工单

   用于记录工单的处理过程和状态。
   - 关联用户。
   - 关联维修人员。
   - 记录工单状态（如待处理、处理中、已完成等）。
   - 记录关于电脑问题的描述。
   - 记录预约时间、到达时间、完成时间。

第 3 章 数据库设计

3.1 数据表设计

根据需求分析和功能设计, 设计如下数据表:

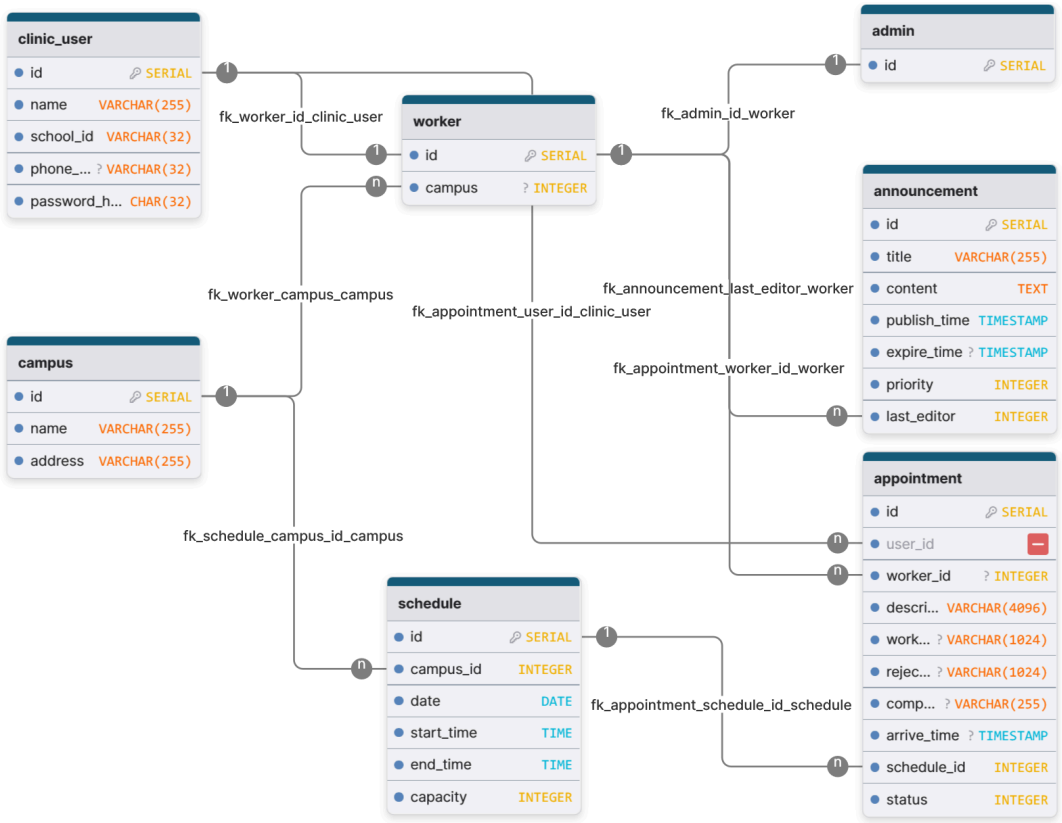

图 3-1 数据表设计

实体属性说明:

- **clinic_user**: 所有用户的基本信息, 包括普通用户和工作人员
- **worker**: 工作人员信息, 继承自用户
- **admin**: 管理员信息, 继承自工作人员
- **campus**: 校区/场地信息
- **schedule**: 诊所开放时间安排
- **appointment**: 预约工单信息
- **announcement**: 公告信息

3.2 ER 图

针对上述需求分析和功能设计, 根据数据表的设计, 绘制出如下的 ER 图:

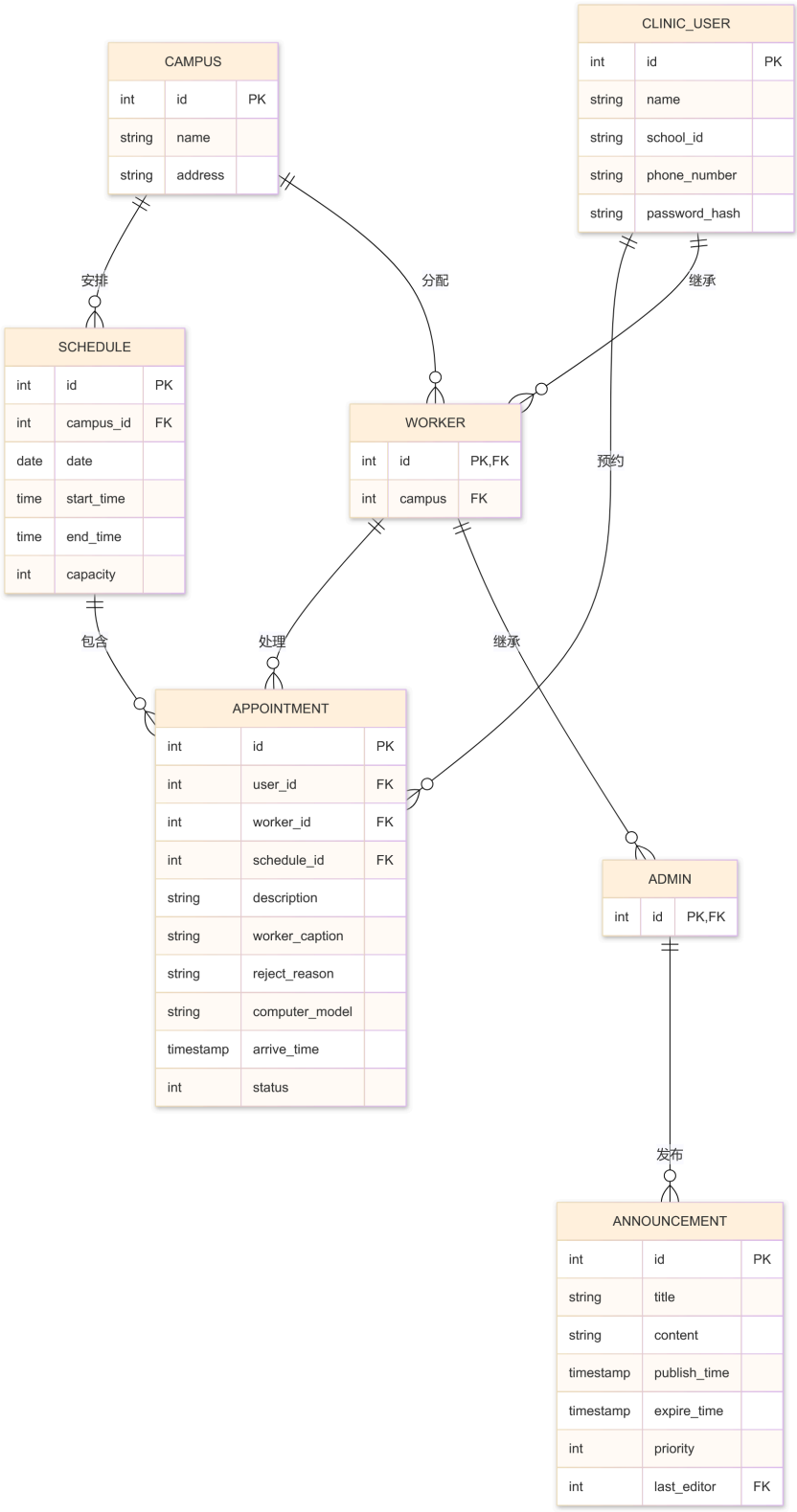

图 3-2 ER 图

在这个 ER 图中, 各实体间的关系如下:

1. **campus** 与 **schedule** 是一对多关系: 一个校区可以有多个日程安排

2. **campus** 与 **worker** 是一对多关系: 一个校区可以有多个工作人员
3. **worker** 继承自 **clinic_user**: 工作人员是用户的特例
4. **admin** 继承自 **worker**: 管理员是工作人员的特例
5. **clinic_user** 与 **appointment** 是一对多关系: 一个用户可以有多个预约
6. **worker** 与 **appointment** 是一对多关系: 一个工作人员可以处理多个预约
7. **schedule** 与 **appointment** 是一对多关系: 一个日程可以有多个预约
8. **worker** 与 **announcement** 是一对多关系: 一个工作人员可以发布多个公告

关键业务约束:

1. 预约状态 (status) 定义了工单的生命周期
2. 日程安排 (schedule) 的开始时间必须早于结束时间
3. 日程安排的容量 (capacity) 必须为正数
4. 当预约状态达到正在处理时, 系统自动记录到达时间(arrive_time)

3.3 范式证明

在本节中，将针对数据库设计进行范式证明，分析每个表是否满足第一范式(1NF)、第二范式(2NF)和第三范式(3NF)。

3.3.1 第一范式(1NF)

第一范式要求关系模式中的每个属性都是不可再分的原子值。形式化定义为：

对于关系模式 $R(U)$，其中 U 是属性集，若 $\forall A \in U$，A 的域中的所有值都是原子的，则称 R 满足第一范式（1NF）。

令 D 表示数据库模式，$R_i \in D$ 表示数据库中的每个关系模式。对于我们的设计：

- **campus 表** $R_1(\text{id}, \text{name}, \text{address})$: $\forall A \in \{\text{id}, \text{name}, \text{address}\}$，$A$ 的值域中的所有值均为原子值，因此 R_1 满足 1NF。
- **clinic_user 表** $R_2(\text{id}, \text{name}, \text{school\_id}, ..., \text{password\_hash})$: 用户表中所有属性均为原子值，因此 R_2 满足 1NF。
- **worker 表** $R_3(\text{id}, \text{campus})$: $\forall A \in \{\text{id}, \text{campus}\}$，$A$ 的值域中的所有值均为原子值，因此 R_3 满足 1NF。
- **admin 表** $R_4(\text{id})$: $\forall A \in \{\text{id}\}$，$A$ 的值域中的所有值均为原子值，因此 R_4 满足 1NF。

- **announcement 表** $R_5(\text{id}, \text{title}, \text{content}, ..., \text{last\_editor})$: 公告表中所有属性均为原子值，因此 R_5 满足 1NF。
- **schedule 表** $R_6(\text{id}, \text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity})$: $\forall A \in \{\text{id}, \text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity}\}$，$A$ 的值域中的所有值均为原子值，因此 R_6 满足 1NF。
- **appointment 表** $R_7(\text{id}, \text{user\_id}, \text{worker\_id}, ..., \text{status})$: 预约表中所有属性均为原子值，因此 R_7 满足 1NF。

因此，$\forall R_i \in D$，R_i 满足 1NF，即整个数据库模式 D 满足第一范式。

3.3.2 第二范式(2NF)

第二范式在第一范式的基础上，要求非主属性完全依赖于主键，而不是部分依赖。形式化定义为：

设 $R(U)$ 是满足 1NF 的关系模式，K 为 R 的候选键，$A \in U$，$A \notin K$ 是非主属性。若对于 R 中的任意非主属性 A，A 完全函数依赖于 R 的每个候选键，则称 R 满足第二范式（2NF）。

用函数依赖表示：不存在 $X \subset K$ 和非主属性 A 使得 $X \to A$。

对于我们的数据库模式 D：

- **campus 表** $R_1(\text{id}, \text{name}, \text{address})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{name}, \text{address}\}$。$\forall A \in \{\text{name}, \text{address}\}$，不存在 $X \subset \{\text{id}\}$ 使得 $X \to A$（因为 id 是单属性，不可再分），所以非主属性完全依赖于主键，满足 2NF。
- **clinic_user 表** $R_2(\text{id}, \text{name}, \text{school\_id}, ..., \text{password\_hash})$: 主键 $K = \{\text{id}\}$，非主属性集合包含所有除 id 外的属性。对所有非主属性，不存在 $X \subset \{\text{id}\}$ 使得 $X \to A$，满足 2NF。
- **worker 表** $R_3(\text{id}, \text{campus})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{campus}\}$。对于 campus，不存在 $X \subset \{\text{id}\}$ 使得 $X \to \text{campus}$，满足 2NF。
- **admin 表** $R_4(\text{id})$: 主键 $K = \{\text{id}\}$，无非主属性，所以平凡地满足 2NF。
- **announcement 表** $R_5(\text{id}, \text{title}, \text{content}, \text{publish\_time}, \text{expire\_time}, \text{priority}, \text{last\_editor})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{title}, \text{content}, \text{publish\_time}, \text{expire\_time}, \text{priority}, \text{last\_editor}\}$。$\forall A \in$

$\{\text{title}, \text{content}, \text{publish\_time}, \text{expire\_time}, \text{priority}, \text{last\_editor}\}$，不存在 $X \subset \{\text{id}\}$ 使得 $X \to A$，满足 2NF。

- **schedule 表** $R_6(\text{id}, \text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity}\}$。 $\forall A \in \{\text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity}\}$，不存在 $X \subset \{\text{id}\}$ 使得 $X \to A$，满足 2NF。
- **appointment 表** $R_7(\text{id}, \text{user\_id}, \text{worker\_id}, ..., \text{status})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K$ 包含所有除 id 外的属性。$\forall A \in$ 非主属性集合，不存在 $X \subset \{\text{id}\}$ 使得 $X \to A$，满足 2NF。

因此，$\forall R_i \in D$，R_i 满足 2NF，即整个数据库模式 D 满足第二范式。

3.3.3 第三范式(3NF)

第三范式在第二范式的基础上，要求非主属性不传递依赖于主键，即非主属性之间不存在函数依赖关系。形式化定义为：

设 $R(U)$ 是满足 2NF 的关系模式，若 R 中不存在这样的非主属性 A 和 B，使得 $A \to B$（即 B 传递函数依赖于键 K），则称 R 满足第三范式（3NF）。

更精确地说，关系模式 $R(U)$ 满足 3NF，当且仅当对于 R 中的每个函数依赖 $X \to A$，要么：

1. X 是超键，或者
2. A 是主属性（即 A 包含于某个候选键中）

对于我们的数据库模式 D：

- **campus 表** $R_1(\text{id}, \text{name}, \text{address})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{name}, \text{address}\}$。我们需要检查是否存在 $\text{name} \to \text{address}$ 或 $\text{address} \to \text{name}$。根据语义，校区名称与地址之间没有函数依赖关系，因此不存在传递依赖，满足 3NF。
- **clinic_user 表** $R_2(\text{id}, \text{name}, \text{school\_id}, \text{phone\_number}, \text{password\_hash})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{name}, \text{school\_id}, \text{phone\_number}, \text{password\_hash}\}$。对于任意 $A, B \in \{\text{name}, \text{school\_id}, \text{phone\_number}, \text{password\_hash}\}$ 且 $A \neq B$，不存在 $A \to B$ 的函数依赖（例如姓名不能决定学号，学号不能决定密码等），因此满足 3NF。
- **worker 表** $R_3(\text{id}, \text{campus})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{campus}\}$。只有一个非主属性，不可能存在传递依赖，满足 3NF。

- **admin 表** $R_4(\text{id})$: 主键 $K = \{\text{id}\}$，无非主属性，所以平凡地满足 3NF。
- **announcement 表** $R_5(\text{id}, \text{title}, \text{content}, ..., \text{last\_editor})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K$ 包含所有除 id 外的属性。 对于任意两个非主属性 A, B，不存在 $A \to B$ 的函数依赖（如标题不能决定内容等），满足 3NF。
- **schedule 表** $R_6(\text{id}, \text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity})$: 主键 $K = \{\text{id}\}$，非主属性集合 $U - K = \{\text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity}\}$。 对于任意 $A, B \in \{\text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity}\}$ 且 $A \neq B$，不存在 $A \to B$ 的函数依赖（例如校区不能决定日期，日期不能决定容量等），因此满足 3NF。
- **appointment 表** $R_7(\text{id}, \text{user\_id}, \text{worker\_id}, ..., \text{status})$: 主键 $K = \{\text{id}\}$，非主属性集合是所有除 id 外的属性。 对于任意两个非主属性 A 和 B，不存在 $A \to B$ 的函数依赖（如用户 ID 不能决定工作人员 ID 等），满足 3NF。

因此，$\forall R_i \in D$，R_i 满足 3NF，即整个数据库模式 D 满足第三范式。

3.3.4 BCNF (Boyce-Codd 范式)

BCNF 是对 3NF 的进一步强化，要求关系模式中所有决定因素必须是候选键。形式化定义为：

设 $R(U)$ 是满足 3NF 的关系模式，若对于 R 中的每个非平凡函数依赖 $X \to A$（其中 $A \notin X$），X 都包含某个候选键，则称 R 满足 BCNF。

换言之，关系模式 $R(U)$ 满足 BCNF，当且仅当对于 R 中的每个函数依赖 $X \to A$，X 必须是 R 的超键。

对于我们的数据库模式 D：

- **campus 表** $R_1(\text{id}, \text{name}, \text{address})$: 主键 $K = \{\text{id}\}$，唯一候选键是 $\{\text{id}\}$。在该表中，$\text{id} \to \text{name}$，$\text{id} \to \text{address}$ 是唯一的非平凡函数依赖， 且 id 是候选键，因此满足 BCNF。
- **clinic_user 表** $R_2(\text{id}, \text{name}, \text{school\_id}, \text{phone\_number}, \text{password\_hash})$: 主键 $K = \{\text{id}\}$，由于 school_id 也被设置为 UNIQUE，因此候选键有 $\{\text{id}\}$ 和 $\{\text{school\_id}\}$。所有函数依赖中的决定因素（id 或 school_id）都是候选键，满足 BCNF。
- **worker 表** $R_3(\text{id}, \text{campus})$: 主键 $K = \{\text{id}\}$，唯一候选键是 $\{\text{id}\}$。 在该表中，$\text{id} \to \text{campus}$ 是唯一的非平凡函数依赖， 且 id 是候选键，因此满足 BCNF。
- **admin 表** $R_4(\text{id})$: 主键 $K = \{\text{id}\}$，只有一个属性，平凡地满足 BCNF。

- **announcement 表** $R_5(\text{id}, \text{title}, \text{content}, ..., \text{last\_editor})$: 主键 $K = \{\text{id}\}$，唯一候选键是 {id}。所有非平凡函数依赖的决定因素都是 {id}，且 {id} 是候选键，满足 BCNF。
- **schedule 表** $R_6(\text{id}, \text{campus\_id}, \text{date}, \text{start\_time}, \text{end\_time}, \text{capacity})$: 主键 $K = \{\text{id}\}$，唯一候选键是 {id}。所有非平凡函数依赖的决定因素都是 {id}，且 {id} 是候选键，满足 BCNF。
- **appointment 表** $R_7(\text{id}, \text{user\_id}, \text{worker\_id}, ..., \text{status})$: 主键 $K = \{\text{id}\}$，唯一候选键是 {id}。所有非平凡函数依赖的决定因素都是 {id}，且 {id} 是候选键，满足 BCNF。

因此，$\forall R_i \in D$，R_i 满足 BCNF，即整个数据库模式 D 满足 Boyce-Codd 范式。

3.3.5 范式设计总结

通过以上严格的数学证明，我们已经验证了本数据库设计完全满足从第一范式到 Boyce-Codd 范式的所有要求。具体而言：

1. **第一范式(1NF)**: $\forall R_i \in D$，$\forall A \in R_i$，A 的值域中的所有值均为原子值，保证了数据的原子性。
2. **第二范式(2NF)**: $\forall R_i \in D$，不存在 $X \subset K$ 和非主属性 A 使得 $X \to A$，消除了非主属性对主码的部分依赖。
3. **第三范式(3NF)**: $\forall R_i \in D$，不存在非主属性 A 和 B 使得 $A \to B$，消除了非主属性之间的传递依赖。
4. **Boyce-Codd 范式(BCNF)**: $\forall R_i \in D$，对于 R_i 中的每个非平凡函数依赖 $X \to A$（其中 $A \notin X$），X 都是超键，确保了所有决定因素都是候选键。

这种严格遵循高级范式的设计方法具有以下优势：

- 最小化数据冗余，节省存储空间
- 消除更新异常（插入异常、删除异常、修改异常）
- 提高数据一致性和完整性
- 增强数据库结构的稳定性和扩展性
- 简化查询优化和索引设计

因此，本数据库模式设计不仅理论上满足规范化要求，在实际应用中也能有效支持业务需求，保证系统的可靠性和性能。

第 4 章 评分

参考评分标准中的得分要求, 对比实际实现, 总结得分情况如下表所示.

表 4-1 得分情况

类别	要求	完成情况
数据库	使用 OpenGauss	✓
	至少 6 张表	✓
前台开发	表的增删改查	✓
	有界面表级联操作	✓
	有应用程序用户权限管理	✓
	采用 Delphi 至少 5 种组件	✓
数据设计	视图	✓
	动态 SQL	✓
	存储过程/函数	✓
	触发器	✓
文档	功能需求、设计说明	✓
	含 ER 图，范式证明	✓
	使用说明书	✓
	源代码、可执行文件	✓
	数据库 SQL 语句	✓
	数据库备份	✓

第 5 章 总结与体会

5.1 项目总结

在本次“电脑诊所管理系统”的设计与实现过程中，我全面应用了数据库设计的理论知识和开发技能，成功构建了一个功能完整、结构合理的数据库应用系统。以下是对项目的总结与体会：

5.1.1 数据库设计成果

1. **需求分析与设计**：基于实际业务需求，设计了一个包含 7 个核心数据表的数据库结构，涵盖用户管理、预约管理、公告管理等功能模块。所有表结构满足第三范式（3NF）甚至更高的 BCNF 范式，确保了数据库的规范性和高效性。
2. **实体关系设计**：构建了清晰的 ER 图，准确表达了各实体间的关系，包括一对多关系和继承关系，如：
   - 校区与日程安排的一对多关系
   - 用户与预约的一对多关系
   - 工作人员继承自用户、管理员继承自工作人员的继承关系
3. **高级数据库功能**：成功应用了数据库的多种高级特性，如：
   - 通过视图简化了复杂查询和数据操作
   - 开发了触发器，实现了业务规则的自动化执行
   - 应用索引，优化查询性能
4. **视图的灵活应用**：

   创建的多个视图（如 appointment_view, worker_view 等）大幅简化了前端应用的数据访问逻辑。特别是 appointment_view，它通过联接多个表，提供了一个全面的预约信息视图，使得前端应用无需编写复杂的 SQL 就能获取完整信息。
5. **触发器的有效利用**：

   通过触发器实现了业务约束，如自动验证日程安排的时间逻辑（开始时间必须早于结束时间）和容量限制（必须为正数），这种方法将业务规则嵌入数据库层，保证了数据一致性和完整性。
6. **用户权限分级设计**：

   系统通过继承关系（普通用户→工作人员→管理员）实现了清晰的权限分级，这种设计既符合业务逻辑，又便于系统权限的管理和扩展。

5.2 收获与体会

通过本次项目的设计与实现，我深刻体会到数据库设计在软件开发中的核心地位。良好的数据库设计不仅是系统稳定性的基础，也是业务逻辑实现的关键。

首先，数据库范式理论在实践中的应用让我认识到，规范的数据结构设计能有效减少数据冗余，提高数据一致性。在项目中应用第三范式和 BCNF 范式的过程，使我对数据依赖关系有了更深入的理解。

其次，数据库高级特性（视图、触发器、索引等）的应用让我体会到，数据库不仅是数据存储的场所，更是业务逻辑的重要载体。通过这些特性，可以将许多业务规则和约束直接实现在数据库层面，简化应用程序的设计。

最后，从 ER 图到物理数据库的转换过程，让我理解了数据库设计的系统性和整体性。一个好的数据库设计应当既满足当前业务需求，又具备面向未来扩展的灵活性。

总的来说，本次项目是理论知识与实践应用的完美结合，不仅巩固了数据库设计的基础知识，也提升了解决实际问题的能力。在未来的工作中，我将继续深化数据库设计与开发的学习，探索更多高级特性的应用，为构建高质量的信息系统打下坚实基础。