

校园生活平台

数据库课程设计检查视频

平台基础 / 功能房预约 / 课程资源分享

数据库原理及应用课程设计

围绕关系模式、完整性约束、索引、触发器、事务和运行验证展开。

讲解路线：整体到局部

视频定位

面向老师检查的单向讲解，重点展示数据库相关的需求、设计、实现和验证。

时间	讲解内容
0:00–1:45	系统整体、用户角色、三个数据库主线
1:45–2:35	从需求到 E-R、关系模式、物理约束的设计链路
2:35–4:00	平台基础：身份、组织、RBAC、审计
4:00–6:15	功能房预约：时间冲突、状态流转、参与人规则
6:15–8:30	课程资源：专业课程、审核流、去重、积分事件
8:30–10:00	运行验证、代码展示顺序和总结

讲解顺序：需求含义 → 数据对象 → 关系模式 → 数据库规则 → 运行结果

三个重点子模块

- 平台基础：身份资料、组织树、RBAC、数据范围、审计日志
- 功能房预约：楼房房间、预约主体、参与人、封禁、时间冲突
- 课程资源分享：专业课程、资源审核、下载事件、积分榜单

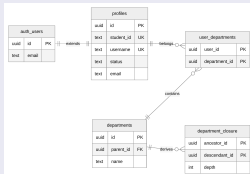
历史保留模块不作为本视频重点，讲解聚焦数据库课程设计主线。

角色与数据库含义

角色	数据库含义
普通用户	资料、部门、角色、预约申请、资源投稿
工作人员	预约审核人、审核时间、驳回原因
管理员	组织、角色、权限、配置、审计
专业负责人	专业范围、资源审核、最佳推荐

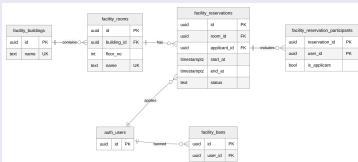
需求到数据库设计链路

主体数据



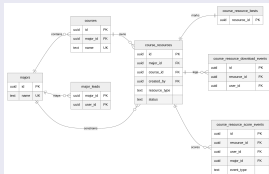
用户、部
专业、课

关系和状态数据



用户角色、角色权限、预约参与人、预约状态、资源审核状态。

事件和审计数据



事件、积分事件、审计日志支撑统计和追溯。

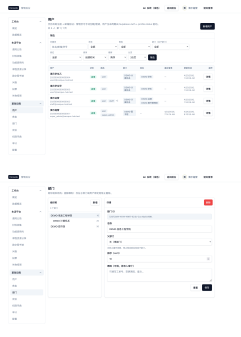
设计原则

主数据边界清晰：必要冗余必须由复合外键、触发器或事务持续控制。

平台基础：身份、组织、权限、审计

数据库设计要点

- `auth.users` 是认证事实源, `profiles.id` 共享主键扩展业务资料。
- `user_roles` 与 `role_permissions` 用复合主键表达多对多授权。
- `department_closure` 支撑本部门及子部门查询, 并由触发器防环维护。
- `audit_logs` 保存操作者快照, 触发器禁止更新和删除。



平台基础：关键数据库对象

```
create table public.user_roles (  
  user_id uuid not null references auth.users(id),  
  role_id uuid not null references public.roles(id),  
  primary key (user_id, role_id)  
);
```

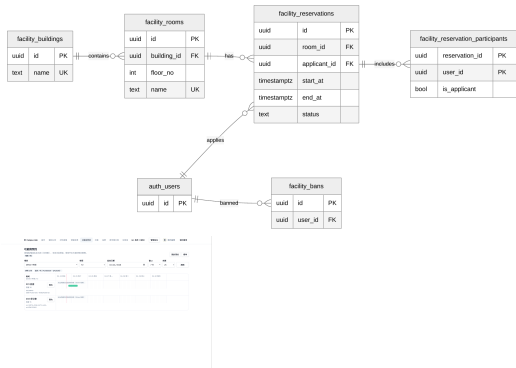
```
create table public.department_closure (  
  ancestor_id uuid not null,  
  descendant_id uuid not null,  
  depth integer not null,  
  primary key (ancestor_id, descendant_id)  
);
```

```
create table public.audit_logs (  
  id uuid primary key,  
  actor_user_id uuid not null,  
  actor_name text,  
  actor_roles jsonb,  
  action text not null,  
  target_type text not null  
);
```

```
create trigger audit_logs_block_update  
before update on public.audit_logs  
for each row execute function  
  public.audit_logs_block_mutation();
```

对应运行验证：用户列表、部门树、角色配置、审计记录。

功能房预约：关系模式与数据库规则



数据库设计要点

- **facility_reservations** 保存房间、申请人、时间、状态、审核和取消字段。
- **facility_reservation_participants** 用复合主键表达预约与用户的联系。
- 同一房间 pending/approved 预约的时间区间不能重叠。
- 状态辅助字段用 CHECK 控制，参与人集合由延迟触发器检查。

功能房预约：排斥约束与事务

```
constraint facility_reservations_room_active_time_excl
exclude using gist (
    room_id with =,
    tstzrange(start_at, end_at, '[)') with &&
)
where (status in ('pending', 'approved'))
```

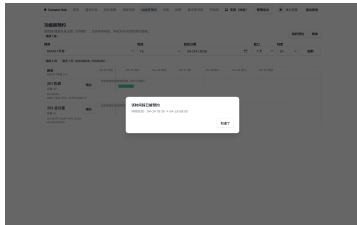
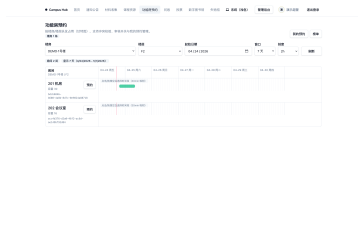
已有开始 < 新结束，且已有结束 > 新开始，表示时间区间重叠。

```
await db.transaction(async (tx) => {
    await lockRoomOrThrow(tx, roomId);
    await assertNoTimeOverlap({
        tx, roomId, startAt, endAt
    });

    const reservationId = await insertReservation(tx);
    await insertParticipants(tx, reservationId, participants);
});
```

创建预约时允许事务中间态，提交阶段再由延迟触发器检查参与人集合。

功能房预约：运行验证



合法预约

时间轴出现占用条，说明预约主体与时间窗口可查询。

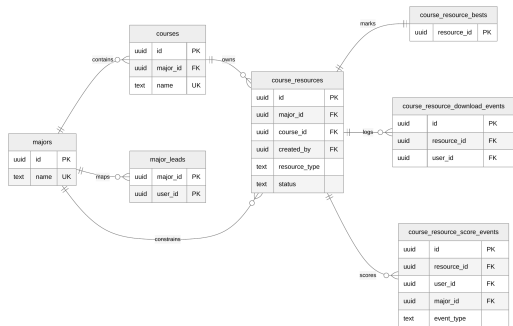
冲突拦截

同房间重叠时间段被拒绝，对应排斥约束。

审核流转

状态字段与审核字段同步变化。

课程资源：教学层级、审核状态、事件事实



数据库设计要点

- majors、courses、major_leads 表达专业、课程和负责人。
- course_resources 保存资源类型、状态、审核字段、发布字段和作者。
- 文件按 sha256 去重，外链按规范化 URL 去重。
- 下载事件和积分事件支撑排行榜与追溯。

课程资源：约束、受控冗余和积分事务

```
constraint course_resources_course_major_fk  
foreign key (course_id, major_id)  
references public.courses(id, major_id);
```

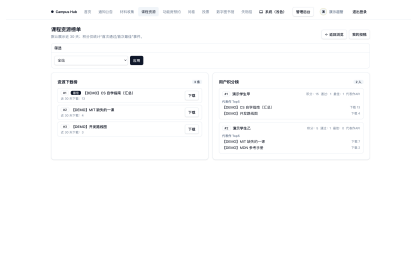
```
create unique index course_resources_course_sha256_active_uq  
on public.course_resources(course_id, sha256)  
where deleted_at is null and resource_type = 'file';
```

major_id 是服务专业范围过滤和统计的受控冗余。

```
await db.transaction(async (tx) => {  
  await tx.update(courseResources)  
    .set({ status: "published", reviewedAt: sql`now()` });  
  
  await tx.insert(courseResourceScoreEvents)  
    .values({ eventType: "approve", delta: approveDelta })  
    .onConflictDoNothing();  
});
```

审核通过和首次积分事件写入在同一事务中完成。

课程资源：运行验证



资源发布详情

课程、专业、类型、审核时间、发布时间、作者和下载次数可回显。

积分与下载榜

下载事件支撑资源榜，积分事件聚合出用户榜。

录制时的代码展示顺序

子模块	展示重点
平台基础	profiles、user_roles、department_closure、audit_logs
功能房预约	排斥约束、参与人延迟触发器、assertNoTimeOverlap()、创建预约事务
课程资源	复合外键、字段互斥、去重索引、积分事件表、审核通过事务
运行验证	用户/部门/角色、预约/冲突/审核、资源发布/积分榜截图

讲代码的原则

不逐行读代码，只说明它把哪个需求规则落实成数据库约束或事务。

总结与后续改进

已完成

需求分析、E-R 建模、关系模式、约束、索引、触发器、事务与运行验证。

主要收获

数据库不仅保存数据，也应主动维护业务规则和统计事实。

后续改进

补充大数据量执行计划；整理视图/物化视图；完善封禁与审计追踪。

谢谢老师观看