

这是一份针对“通用航空飞行计划审批平台”的详细系统设计方案。构建了以下四个核心部分：
实体关系图 (ERD)、数据流图 (DFD)、关系模式以及物理存储结构安排。

1. 实体关系图 (ERD) - 概念模型

这部分描述了系统中各个实体及其相互关系。主码为红色字体、外码为斜体

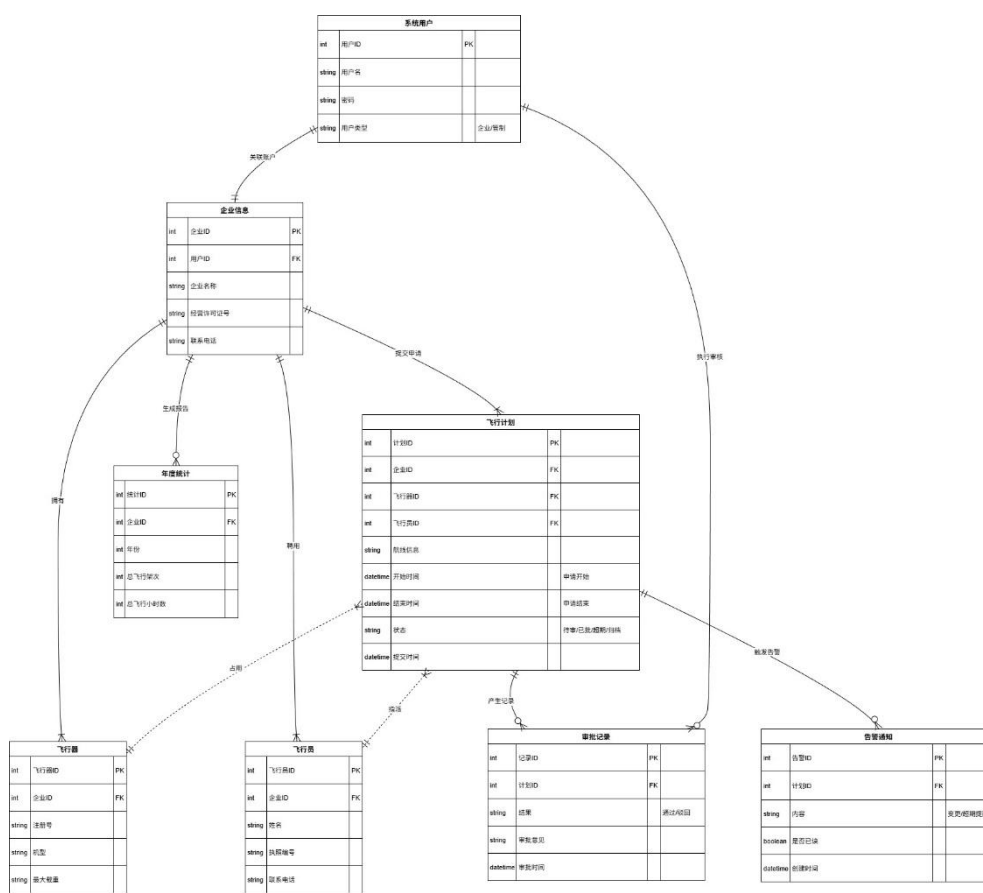
核心实体：

- 用户 (User): 分为“通航运营企业”和“管制人员”。
属性包括：用户 ID、用户名、密码、用户类型
- 企业信息 (Enterprise Profile): 运营企业的详细资质信息。
属性包括：企业 ID、用户 ID、企业名称、经营许可证号、联系电话
- 飞行器 (Aircraft): 直升机或小型飞机，隶属于企业。
属性包括：飞行器 ID、企业 ID、注册号、机型、最大载重
- 飞行员 (Pilot): 隶属于企业。
属性包括：飞行员 ID、企业 ID、姓名、执照编号、联系电话
- 飞行计划 (Flight Plan): 核心业务对象，包含航线、时间等。
属性包括：计划 ID、企业 ID、飞行器 ID、飞行员 ID、航线信息、开始时间、结束时间、状态、提交时间
- 审批记录 (Approval Record): 管制人员对计划的批复。
属性包括：记录 ID、计划 ID、结果、审批意见、审批时间
- 告警/通知 (Alert/Notification): 变更提醒、超时报警。
属性包括：告警 ID、计划 ID、内容、是否已读、创建时间
- 年度统计：
属性包括：统计 ID、企业 ID、年份、总飞行架次、总飞行小时数

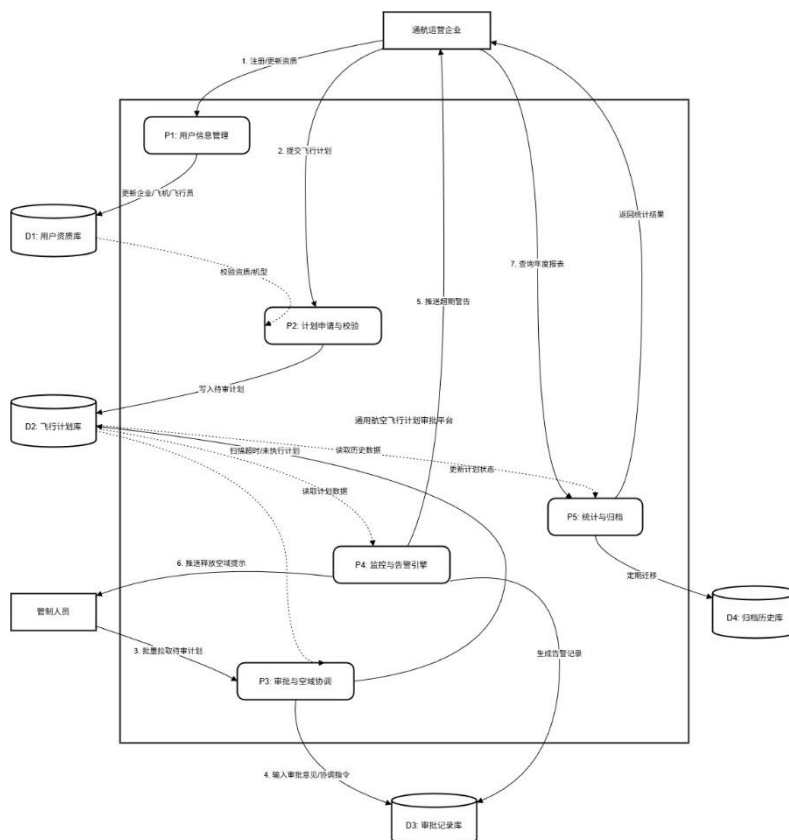
实体关系描述：

1. 企业 - 拥有 -> 飞行器 (1:N)
2. 企业 - 聘用 -> 飞行员 (1:N)
3. 企业 - 提交 -> 飞行计划 (1:N)
4. 飞行计划 - 关联 -> 飞行器 (1:1, 一次计划对应一架飞机)
5. 飞行计划 - 关联 -> 飞行员 (1:1, 一次计划对应一名主飞)
6. 管制人员 - 审核 -> 飞行计划 (1:N)

7. 飞行计划 - 生成 -> 告警/通知 (1:N)



2. 数据流图 (DFD) - 逻辑模型



这部分描述数据如何在系统中流动。

顶层图 (Context Diagram)

- **外部实体:** 通航运营企业、管制人员。
- **核心处理:** 通用航空飞行计划审批平台。
 - 企业 -> 输入计划申请、基本信息 -> 平台
 - 平台 -> 输出审批结果、变更提醒、统计报表 -> 企业
 - 平台 -> 输出待审计划、超期报警 -> 管制人员
 - 管制人员 -> 输入审批意见 (含批量)、空域协调指令 -> 平台

0 层图 (Level-0 DFD - 功能分解)

1. **P1 用户与基础信息管理:**
 - 输入: 企业注册信息、飞机/飞行员资料。
 - 存储: 更新用户库、航空器库、飞行员库。
2. **P2 飞行计划管理 (申请/变更):**
 - 输入: 机型、飞行员、航线、申请时段。
 - 处理: 格式校验、冲突检测。
 - 输出: 生成待审核计划数据。
3. **P3 审批与空域协调:**
 - 输入: 待审计划列表。
 - 处理: 管制员审核 (支持批量)、资源协调。
 - 输出: 更新计划状态 (已批准/驳回)、生成批复记录。
4. **P4 监控与告警引擎:**
 - 输入: 当前时间、计划执行状态。
 - 处理: 扫描超期未执行计划、计划变更监控。
 - 输出: 触发报警信号、发送通知给相关人员。
5. **P5 统计与归档:**
 - 输入: 历史计划数据。
 - 处理: 年度统计计算、过期数据迁移。
 - 输出: 年度统计报表、归档数据。

3. 关系模式 (Relational Schema) - 逻辑结构

基于需求设计的数据库表结构 (主键用斜体标识, 外键标为 *FK*)。

A. 基础信息类

1. **用户表 (Sys_User)**
 - 字段: (*UserID*, Username, Password, UserType, CreatedTime)
 - 注: *UserType* 区分是企业用户还是管制人员
2. **运营企业表 (Enterprise_Info)**
 - 字段: (*EnterpriseID*, *UserID FK*, CompanyName, LicenseCode, ContactPhone)
3. **飞行器表 (Aircraft_Info)**
 - 字段: (*AircraftID*, *EnterpriseID FK*, RegNumber, ModelType, MaxLoad)
 - 对应需求: 机型信息
4. **飞行员表 (Pilot_Info)**
 - 字段: (*PilotID*, *EnterpriseID FK*, PilotName, LicenseNo, Phone)

B. 核心业务类

5. **飞行计划表 (Flight_Plan)**

- 字段: (*PlanID*, *EnterpriseID FK*, *AircraftID FK*, *PilotID FK*, *RouteDescription*, *StartTime*, *EndTime*, *FlightType*, *Status*, *SubmitTime*, *ReviewerID FK*)
- 注: *Status* 包含 (待审核, 已批准, 已驳回, 已完成, 超期未执行, 已归档)
- 对应需求: 航线、申请时段、机型、飞行员

6. 审批记录表 (Approval_Log)

- 字段: (*LogID*, *PlanID FK*, *ReviewerID FK*, *Result*, *Comments*, *ApprovalTime*)
- 对应需求: 批量下发批复

C. 辅助功能类

7. 告警通知表 (Alert_Notification)

- 字段: (*AlertID*, *PlanID FK*, *ReceiverID FK*, *MessageContent*, *IsRead*, *AlertType*, *CreatedTime*)
- 对应需求: 变更提醒、超期报警

8. 年度统计表 (Annual_Stats)

- 字段: (*StatID*, *Year*, *EnterpriseID*, *TotalFlights*, *TotalHours*, *ApprovedCount*)
- 对应需求: 飞行计划年度统计

4. 存储安排的物理结构 (Physical Structure)

为了满足需求中提到的“数据归档”、“年度统计”以及“批量处理”的高效性，物理存储设计建议如下：

A. 索引策略 (Indexing Strategy)

为了加快查询和审批速度：

- **聚集索引 (Clustered Index):** 在所有表的主键 (ID) 上建立。
- **非聚集索引 (Non-Clustered Index):**
 - *Flight_Plan* 表：在 *Status* (状态) 字段建立索引，方便管制员快速拉取“待审核”列表。
 - *Flight_Plan* 表：在 *StartTime* 和 *EndTime* 建立索引，用于快速检索“申请时段”和系统后台扫描“超期未执行”的任务。
 - *Flight_Plan* 表：在 *EnterpriseID* 建立索引，方便企业查询自己的历史记录。

B. 数据分区 (Partitioning)

考虑到飞行计划数据会随时间累积，且需求明确要求“数据归档”：

- **范围分区 (Range Partitioning):** 对 *Flight_Plan* 表按 *SubmitTime* (提交时间) 进行按年或按月分区。
 - 优势: 生成“年度统计”时，只需扫描特定分区，极大提高速度；旧数据归档时，可以直接卸载旧分区，便于管理。

C. 存储介质与冷热分离

- **热数据 (Hot Data):** 当前年份的、状态为“待审核/执行中”的计划。
 - 存储位置: 高性能 SSD 存储阵列，保证高并发下的读写速度。
- **冷数据 (Cold Data/Archived):** 超过 1 年的、状态为“已归档”的历史数据。
 - 存储位置: 大容量 HDD 或云存储归档桶。满足需求中的“数据归档”功能，降低成本。

D. 安全与备份

- **RAID 配置:** 数据库服务器采用 RAID 10，兼顾读写性能与数据冗余安全性。
- **异地容灾:** 由于涉及空域资源协调，建议设置每日增量备份和每周全量备份，并异地存储。

超期警报：

针对“超期未执行报警”这一关键功能，从业务逻辑、处理流程、数据状态变迁以及技术实现机制四个维度进行详细拆解。

这个功能的核心在于自动化监控，确保管制人员和企业能及时获知飞行计划的异常状态，避免空域资源的浪费或安全隐患。

1. 业务逻辑定义 (Business Logic)

首先，我们需要定义什么是“超期未执行”。通常在通航领域，逻辑如下：

- **前提条件：** 飞行计划的状态必须是“已批准 (Approved)”。
- **触发条件：** 当前系统时间 > (计划结束时间 + 容忍阈值)。
 - **容忍阈值 (Threshold):** 例如 30 分钟或 1 小时。如果计划结束时间是 14:00, 到了 14:30 飞机仍未报告起飞或结束，且未申请延期，即判定为超期。
- **执行动作：**
 1. 将计划状态变更为“超期未执行”。
 2. 向运营企业发送警告（短信/App 推送）。
 3. 向管制员发送高亮报警，提示释放空域资源。

2. 详细数据流程图 (DFD - 局部细化)

这是针对“P4 监控与告警引擎”的详细展开：

1. **定时触发器 (Timer):** 每隔特定时间（如 5 分钟）发出一个脉冲信号。
2. **扫描进程 (Scanner):**
 - 读取 Flight_Plan 表。
 - 筛选条件: Status = '已批准' AND EndTime < (CurrentTime - 30min).
3. **状态更新器 (Updater):**
 - 将筛选出的计划 Status 更新为 '超期未执行'。
4. **告警生成器 (Notifier):**
 - 向 Alert_Notification 表写入两条记录：
 - 一条发给企业（内容：您的计划 XX 已超期，请说明原因）。
 - 一条发给管制员（内容：计划 XX 未执行，空域已释放）。

3. 数据状态变迁与存储设计

A. 状态机 (State Machine)

在这个模块中，飞行计划的状态流转如下：

待审核 $\xrightarrow{\text{通过}}$ 已批准 $\xrightarrow{\text{未起飞/超时检测}}$ 超期未执行 $\xrightarrow{\text{归档}}$ 历史记录

B. 涉及的数据库操作 (逻辑描述)

虽然不需要 SQL 代码，但逻辑操作如下：

1. 读取 (Read):

利用物理结构设计中的非聚集索引（针对 Status 和 EndTime），系统快速定位数据：
查找所有状态为“已批准”且结束时间小于当前时间减去阈值的记录。

2. 写入 (Write) - 事务处理:

这是一个原子操作，必须同时完成以下几步，否则回滚：

- **Step 1:** 更新 Flight_Plan 表，Status 设为 EXPIRED。

- **Step 2:** 插入 Alert_Notification 表, ReceiverID = 企业 ID。
 - **Step 3:** 插入 Alert_Notification 表, ReceiverID = 管制员 ID。
-

4. 物理实现机制 (Physical Implementation)

为了保证系统性能, 不建议由前端用户操作触发检查, 而是采用后端**后台守护进程**。

A. 轮询机制 (Polling Strategy)

- **组件:** 使用服务器端的定时任务调度器 (如 Linux Cron Job 或 Java Quartz)。
- **频率:** 建议设置为 **每 5-10 分钟** 执行一次。太频繁浪费资源, 太慢则反应迟钝。

B. 性能优化 (针对物理结构)

由于随着时间推移, 历史计划会非常多, 为了防止扫描速度变慢, 利用之前设计的**分区 (Partitioning)**:

- 扫描程序只扫描**“当前月份”或“当前年份”**的分区。
 - 这样即使历史库里有 100 万条数据, 扫描程序也只需要在几千条“活跃数据”中查找, 效率极高。
-

5. 用户界面交互设计 (UI/UX)

对于运营企业 (用户端):

- **推送通知:** 手机会收到弹窗: “您的飞行计划 [FP-20231001] 已超期并未执行, 系统已自动取消。”
- **列表显示:** 在“我的计划”列表中, 该条目变成灰色或红色, 状态标签显示为“超期”。

对于管制人员 (管理端):

- **Dashboard 高亮:** 管理后台首页通常有一个“异常监控面板”。
- **声光报警:** 如果是实时监控大屏, 超期的条目会**闪烁红色**, 并可能伴随提示音, 提醒管制员该空域时段并未被占用, 可以重新分配给其他紧急任务。

