**审批功能模块设计及思考**

**业务背景**

审批功能的需求起源于财务部门，财务的日常工作，涉及到对账转账相关。钱对于一个公司来说，是一个很敏感的东西，支出需要走非常严谨的流程。需要随着业务量的日趋壮大，外部供应商之多，传统线下各类单据逐级审批的方式是一种耗时、耗力的低效工作，迫切需要能通过走线上操作的方式，去提升日常的工作效率。

**需求整理**

审批功能要完成的就是：不同的单据，可以配置不等数量的不同过程的审批流程；对于同一个类型的单据来说，不同的审批流程配置又可以指定不同身份(诸如：按部门、按角色、按员工等)，并且每一级审批可以配置是否可跨级(比如：一个单据按次序，配置了三级审批，从低到高，分别是A、B、C，其中A已经审批完成，B设置为允许跨级审批，即意味着在B不审批的情况下，C可以直接审批，使整个单据达到最终审批通过的状态；反之，如果B设置为不可跨级审批，那么C这时候操作单据，是没有权限的)。我们这里说的审批，其实有点跟常见的OA系统里的离职申请审批有类似之处，不同的是，OA系统似乎没有可越级审批的设置，并且每一级的审批权限都落实到具体的人，而没有我们上面提到的可设置按不同的审批身份**。**因为我们的审批功能是跟不同的单据有间接关系的，所以也会有和不同单据逻辑耦合的可能，比如状态回写、以及更复杂的下游逻辑。

**数据库设计**

　　所谓的“流程审批”，就是一份数据，在设定好的节点之间流转。在流转过程中，数据可能发生变化。

　　对于一个流程，它的节点和连线是固定的。通常是由用户公司的管理员通过二次开发工具设定流程。

　　要做一个通用的流程系统，大概应建立如下数据表：

　　1. 流程表（tbl\_flow）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| flow\_id | int | 流程编号，主键 |
| flow\_no | int | 流程号，唯一列 |
| flow\_name | varchar2(100) | 流程名称 |
| remark | varchar2(500) | 备注 |

　　示例数据：

|  |  |  |  |
| --- | --- | --- | --- |
| **flow\_id** | **flow\_no** | **flow\_name** | **remark** |
| 1 | 5 | 请假流程 | 请假流程 |
| 2 | 10 | 报销流程 | 报销流程 |

　　2. 流程节点表（tbl\_flow\_node）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| flow\_node\_id | int | 流程节点编号，主键 |
| flow\_no | int | 流程号，与流程表对应 |
| flow\_node\_name | varchar2(100) | 流程节点名称 |
| flow\_node\_role | varchar2(100) | 流程角色 |
| flow\_node\_role\_userid | int | 节点负责人用户ID |
| remark | varchar2(500) | 备注 |

　　示例数据：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **flow\_node\_id** | **flow\_no** | **flow\_node\_name** | **flow\_node\_role** | **remark** |
| 1 | 5 | 提交请假单 | 无 | 提交请假单 |
| 2 | 5 | 部门经理审批 | 部门经理 | 部门经理审批 |
| 3 | 5 | 总经理审批 | 总经理 | 总经理审批 |

　　3. 流程线表（tbl\_flow\_line）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| flow\_line\_id | int | 流程线编号，主键 |
| flow\_no | int | 流程号，与流程表对应 |
| prev\_node\_id | int | 前一节点编号 |
| next\_node\_id | int | 后一节点编号 |
| remark | varchar2(500) | 备注 |

　　示例数据：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **flow\_line\_id** | **flow\_no** | **prev\_node\_id** | **next\_node\_id** | **remark** |
| 1 | 5 | 1 | 2 | 提交部门经理审批 |
| 2 | 5 | 2 | 3 | 提交总经理审批 |

　　4. 流程角色\_员工表（tbl\_flow\_role\_user）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| flow\_role\_name | varchar2(100) | 流程角色名称 |
| user\_id | int | 员工编号 |
| dept\_id | int | 部门编号 |

　　示例数据：

|  |  |  |
| --- | --- | --- |
| **flow\_role\_name** | **user\_id** | **dept\_id** |
| 部门经理 | 13 | 2 |
| 部门经理 | 26 | 4 |
| 总经理 | 3 | 1 |

　　以上就是一个通用的审批流程所需要的数据表。

**具体实现**

这里，我们需要先强调一下我们的审批功能要解决的一个最复杂审批链场景：首先，所有类型的单据都需要发起审批，只有发起审批后的单据才能进入逐级审批的操作流程，在理想的情况下，发起审批后逐级审批，审批流中的所有节点都审批通过，才认为整个单据审批最后通过，任何一个审批级别审批标志为“不通过”，都视为整个审批不通过（中间审批状态、最终审批状态，都需要更新各单据主表的审批状态，以及可能的各类单据的个性化业务逻辑）；如果某个审批级别审批标志为“驳回”，那么单据需要重新再发起审批，发起后的业务逻辑复用初始化时候的逻辑。那么，

1.如何确定什么时候是需要更新单据的最终审批的成功状态？

单据的最终审批通过，是需要审批节点上所有审批级别都是审批通过的（当然，可跨级审批的节点除外），我们以表3该单据的最近一条审批进度记录作为参考基准，如果最近的一条审批状态是审批通过的，且当前操作是可进行最后一级审批节点操作的，如果操作为“审批通过”，我们就可以认为整个单据是审批通过了。总结来说，在这个过程中，我们一直在依赖“最近一条审批进度记录”,它是否可靠，是整个模块的关键。

2.审批功能模块在业务逻辑上，最复杂的是什么？

是结合某个单据当下最近审批进度决定的当前审批人是否具备审批的权限，审批权限的判断最终又落实到当前用户是否具有下级审批级别（如果有）的操作权限，或者如果下级审批是可越级的，就需要判断是否具备下下级（如果有）的操作权限，以此类推，这里就需要一个递归的判断过程...所以我们将权限的校验逻辑封装在一个块里，作为一个整体对调用方（仍然是在审批模块范畴）透明，涉及权限校验的排查、迭代，我们只需要维护这块代码。

**一些思考**

1.模块设计之初，是希望是高内聚的，并且与各类单据自身逻辑是低耦合的，但在具体编码实现的时候，没有完全隔离单据的个性化逻辑，但目前来说，如果迭代的话，耦合逻辑抽离出来不算麻烦。

2.出现了分布式事务一致性的问题。上面提到了，我们在审批的逻辑里，夹杂了更新各类单据的审批状态，目前在当前业务场景和需求严谨性考虑下，我们只保证了审批主流程顺畅，单据审批状态的更新结果没有考虑。也就是说，审批模块的审批记录结果和单据的审批状态没有最终一致性关联。所以，我们需要对要求比较高的单据做强一致性关联；弱要求的单据做最终一致性关联。强一致性关联上，我们可能需要考虑引入诸如分布式事务一致性解决方案中的两（三）阶段提交等方案；最终一致性关联可能需要我们有事务补偿机制。这些都是要根据业务需要，做的思考，需要单独做讨论的问题，宗旨是复杂的事情简单做。