|  |
| --- |
| 一丹软件 |
| 一丹电子病历时限质量控制 |
| 需求说明及解决方案 |
|  |
| **武文俊** |
| **2013-1-4** |

|  |
| --- |
| 实现电子病历时限质量管控的机制，同时作为病历质量的评分依据 |

目录

[1、 系统需求简述 3](#_Toc345067564)

[2、分析 3](#_Toc345067565)

[2.1难点 3](#_Toc345067566)

[3、解决方案 4](#_Toc345067567)

[3.1数据库设计 4](#_Toc345067568)

[3.2.项目代码要点 7](#_Toc345067569)

[3.2.1 解决方案总体框架 7](#_Toc345067570)

[3.2.2 部分算法说明 8](#_Toc345067571)

[4.开发计划及工时分配 12](#_Toc345067572)

[5.进度监控 13](#_Toc345067573)

# 系统需求简述

病历是指医务人员在医疗活动过程中形成的文字、符号、图表、影像、切片等资料的总和，包括门(急)诊病历和住院病历。病历书写是指医务人员通过问诊、查体、辅助检查、诊断、治疗、护理等医疗活动获得有关资料，并进行归纳分析、整理形成医疗活动记录的行为。

病历书写的基本原则：病历书写应当客观、真实、准确、及时、完整。其中“及时性”就表现在病历的时限上，即病历的时间限制。

本功能用于决定发出病历需要书写的提醒和警告信息，以及病历时限的相关控制操作。

每份病历需要书写的时间标准不一样，而且不是每份病历都是需要书写的，病历的书写需要满足一定的条件，所以这里使用配置的方式控制病历的时限控制。

病历时限触发规则如下：

（1） 一次性，即只触发一次，下次再触发也不执行

（2） 触发一次执行一次

（3） 循环触发，分为无限循环和有限循环 1~Max

# 2、分析

## 2.1难点

整个系统病历时限的触发分为4大块，分别是基础数据维护、定时服务触发和其他功能模块的调用。

内部触发器触发又分为3种病人状态改变、医嘱改变、病历改变，需要根据不同的触发条件进行相应的处理，保证病历时限控制的实时性。

定时服务触发是使用Job的方式定时运行触发，需要考虑到性能和并发的问题，使系统不仅要保证正确性，也要保证一定的速度，所以需要规划好系统内部操作的流程。不能因为这个功能而影响到整个系统的性能。

在设置时限规则的时候要保证规则的可配置性，这样就需要考虑整个系统的灵活性，保证系统的全面性，出现新需求时只需要配置一下时限规则即可，从而达到不需要修改代码或少修改代码的目的。

# 3、解决方案

## 3.1数据库设计

#### 3.1.1质量控制规则库(QCRULE)

**功能:**

1. 保存时限控制的规则库
2. 决定病历质量控制记录库QCRecord中的数据

**表结构:**

|  |  |  |
| --- | --- | --- |
| **质量控制规则库 QCRULE** | | |
| RULECODE | VARCHAR2(64) PK | 规则代码 |
| CONDITIONCODE | VARCHAR2(64) | 规则条件代码(QCCondition.Code) |
| ~~RESULTCODE~~ | ~~VARCHAR2(64)~~ | ~~规则结果代码(QCResult.Code)~~ |
| DESCRIPTION | VARCHAR2(64) | 规则描述 |
| REMINDER | VARCHAR2(255) | 提示信息 |
| FOULMESSAGE | VARCHAR2(255) | 违规信息 |
| DELAYTIME | NUMBER(14,4) | 延迟时间(以秒计算) |
| TIMELIMIT | NUMBER(14,4) | 规则时限(以秒计算) |
| ~~RELATEDRULE~~ | ~~VARCHAR2(1024)~~ | ~~----------------无效----------------~~ |
| ~~RELATEDMARK~~ | ~~INTEGER~~ | ~~----------------无效----------------~~ |
| MARK | INTEGER | 操作方式 0:一次性 1: 触发一次执行一次 2:循环 |
| CYCLETIMES | INTEGER | 循环处理时的次数(0=无穷) MARK==2时有值，否则为NULL |
| CYCLEINTERVAL | NUMBER(14,4) | 循环间隔 (单位:秒) MARK==2时有效，否则为NULL |
| DOCOTORLEVEL | INTEGER | 医生级别 CategoryDetail.CategoryID = 20  特殊的:0表示全部级别)  2000 主任医师  2001 副主任医师  2002 主治医师  2003 住院医师  2004 护士 |
| SORTCODE | VARCHAR2(64) | 规则分类代码(RuleCategory.Code) |
| VALID | INTEGER | 有效标志 0:无效 1:有效 |
| MEMO | VARCHAR2(64) | 备注 |
| [SCORE](http://dict.baidu.com/s?wd=score) | NUMBER(14,4) | 扣分 |
| QCCODE | VARCHAR2(64) | 监控代码 EMRQCITEM.ID |

#### 3.1.2质量控制条件库(QCCONDITION)

**功能:**

1. 控制病历时限的触发时机
2. 决定病历质量控制记录库QCRecord中的数据

**表结构:**

|  |  |  |
| --- | --- | --- |
| **质量控制条件库 QCCONDITION** | | |
| CODE | VARCHAR2(64) PK | 条件代码 |
| ~~CATEGORY~~ | ~~INTEGER~~ | ~~条件分类 CategoryDetail.CategoryID = 39~~  ~~3901 病人状态改变~~  ~~3902 病历记录改变~~  ~~3903 医嘱记录改变~~ |
| DESCRIPTION | VARCHAR2(64) | 条件描述 |
| TABLENAME | VARCHAR2(32) | 需要配置的表名 |
| COLUMNNAME | VARCHAR2(32) | TABLENAME表中对应的列名 |
| COLUMNVALUE | VARCHAR2(32) | TABLENAME表中COLUMNNAME对应列的具体值 |
| TIMECOLUMNNAME | VARCHAR2(32) | COLUMNNAME对应的时间字段名 |
| TIMERANGE | NUMBER(14,4) | TIMECOLUMNNAME对应的时间范围(以秒计算) |
| PATNOCOLUMNNAME | VARCHAR2(32) | TABLENAME表中病人序号字段 |
| DBLINK | VARCHAR2(32) | TABLENAME所在的数据库 分为：EMRDB、HISDB |
| ~~CONDITION~~ | ~~VARCHAR2(1024)~~ | ~~条件设置~~ |
| ~~TIMEINSTALL~~ | ~~VARCHAR2(1024)~~ | ~~时间设置~~ |
| VALID | INTEGER | 有效标志 0:无效 1:有效 |
| MEMO | VARCHAR2(64) | 备注 |

#### ~~3.1.3质量控制结果库(QCRESULT)~~

**~~功能:~~**

1. ~~本表用于控制规则时限中病历对应的监控代码~~
2. ~~用于关闭时限规则~~

**~~表结构:~~**

|  |  |  |
| --- | --- | --- |
| **~~质量控制结果库 QCRESULT~~** | | |
| ~~CODE~~ | ~~VARCHAR2(64) PK~~ | ~~结果代码~~ |
| ~~CATEGORY~~ | ~~INTEGER~~ | ~~结果分类 CategoryDetail.CategoryID = 40~~  ~~4001 病历记录改变~~  ~~4002 时间提示有效期~~ |
| ~~DESCRIPTION~~ | ~~VARCHAR2(64)~~ | ~~结果描述~~ |
| ~~RESULT~~ | ~~VARCHAR2(1024)~~ | ~~结果设置~~ |
| ~~TIME~~ | ~~VARCHAR2(1024)~~ | ~~时间设置~~ |
| ~~VALID~~ | ~~INTEGER~~ | ~~有效标志 0:无效 1:有效~~ |
| ~~MEMO~~ | ~~VARCHAR2(64)~~ | ~~备注~~ |
| ~~QCCODE~~ | ~~VARCHAR2(64)~~ | ~~监控代码~~ |

~~QCCode字段的数据源：EMRQCITEM表~~

#### 3.1.4 质量控制规则分类库(RULECATEGORY)

**功能：**

1. 用于对QCRule中的时限规则进行分类

**表结构:**

|  |  |  |
| --- | --- | --- |
| **质量控制规则分类库 RULECATEGORY** | | |
| CODE | VARCHAR2(64) PK | 规则分类代码 |
| DESCRIPTION | VARCHAR2(64) | 规则分类描述 |
| PY | VARCHAR2(8) | 拼音 |
| WB | VARCHAR2(8) | 五笔 |
| VALID | INTEGER | 有效标志 0:无效 1:有效 |
| MEMO | VARCHAR2(64) | 备注 |

针对QCRule表中的记录进行分类的基础数据

#### 3.1.5病历质量控制记录库(QCRECORD)

功能:

1. 保存了具体病人对应的时限信息
2. 显示病历时限的提示和警告信息

|  |  |  |
| --- | --- | --- |
| **病历质量控制记录库 QCRECORD** | | |
| ID | NUMBER(12) PK | 序号 |
| NOOFINPAT | NUMBER(9) | 首页序号 Inpatient.NoofInpat |
| NOOFRECORD | NUMBER(12) | 病历号 Inpatient.Noofrecord |
| RULECODE | VARCHAR2(64) | 规则代码 QCRule. RULECODE |
| CONDITIONCODE | VARCHAR2(64) | 条件代码(规则的前提条件) QCCondition.Code |
| CONDITIONTIME | CHAR(19) | 条件时间， 即触发的时间 |
| REALCONDITIONTIME | CHAR(19) | 条件时间(条件成立的时间) 触发的时间 + 延迟时间 |
| CONDITION | INTEGER | 条件状态(是否完成) 0:未完成 1:完成  对于条件状态未完成的记录在系统的界面中不予显示 |
| ~~RESULTCODE~~ | ~~VARCHAR2(64)~~ | ~~结果代码(QCResult.Code)~~ |
| RESULTTIME | CHAR(19) | 结果时间(结果成立的时间) |
| RESULT | INTEGER | 结果状态(是否完成) 0:未完成 1:完成 |
| FOULSTATE | INTEGER | 违规状态(是否违规) 0:不违规 1:违规 |
| REMINDER | VARCHAR2(255) | 提示信息 |
| FOULMESSAGE | VARCHAR2(255) | 违规信息 |
| OPERATETIME | CHAR(19) | 操作时间，即记录创建的时间 |
| ~~DOCOTORID~~ | ~~VARCHAR2(6)~~ | ~~医生代码----------------无效----------------~~ |
| ~~CRDOCOTORID~~ | ~~VARCHAR2(6)~~ | ~~创建职工代码----------------无效----------------~~ |
| DOCOTORLEVEL | INTEGER | 责任医生级别QCRule.DOCOTORLEVEL |
| ISCYCLE | INTEGER | 是否循环 0:不循环 1:循环 |
| FIRSTCYCLERECORDID | VARCHAR2(64) | 循环规则的第一条记录ID QCRECORD.ID，ISCYCLE==1时生效 |
| CYCLETIMES | INTEGER | 循环计数(用于限定次数的时限规则) ，ISCYCLE==1时生效 |
| VALID | INTEGER | 有效标志 0:无效 1:有效 |
| MEMO | VARCHAR2(64) | 备注 |
| TIMELIMIT | INTEGER | 规则时限(以秒计算) |
| QCCODE | VARCHAR2(64) | 监控代码 |
| RECORDDETAILID | INTEGER | 病历编号 Recorddetail.ID |
| [SCORE](http://dict.baidu.com/s?wd=score) | NUMBER(14,4) | 扣分 |
| ISSTOPCYCLE | INTEGER | 针对循环时限规则，即ISCYCLE == 1，表示下一次是否需要插入初始化数据到QCRecord表中，即是否停止循环  ISCYCLENEXT == 1， 表示下一次不需要插入数据，即循环停止 |

1. Reminder、FoulMessage、DoctorLevel、ISCYCLE 、CycleTimes、TimeLimit、Score从QCRule表获得QCCode通过QCRule.ResultCode = QCResult.Code得到QCResult.QCCode
2. 判断延迟时间：

延迟时间 > 0

如果 触发时间 <= 当前时间 < 触发时间 + 延迟时间，则Condition = 0

如果 触发时间 + 延迟时间 <= 当前时间，则Condition = 1

延迟时间 = 0

如果 触发时间 <= 当前时间，则 Condition = 1

1. 只有当Condition == 1（条件已完成）AND Result == 0（病历未完成）的时候，才会出现需要填写病历的提醒

## 3.2.项目代码要点

### 3.2.1 解决方案总体框架

#### 3.2.1.1 应用管理部分



### 3.2.2 部分算法说明

分为主动和被动两种方式：

主动方式

优点：只需要进行配置，无需进行外部触发或触发器触发，功能模块间的耦合性比较小

缺点：（1）不能保证数据的实时性，有一定的延迟

（2）由于没有外部触发，所有每次遍历都需要大范围的判断，如果业务逻辑复杂，对系统的性能会造成一定的影响。

（3）会出现如下的情况：由于系统是定时运行，如果在运行间隔病人状态变化多次，导致系统只能对其最后一次的状态进行处理

（4）所有逻辑均在内部判断，导致其逻辑较为复杂，维护起来难度较高

**被动方式：**

此功能核心模块主要分为三部分：

1. **生成病历时限的提醒信息，根据时限的条件判断，插入初始数据到QCRecord表。**

触发条件：病人状态改变、医嘱改变、病历改变

例如：

病人状态改变：病人入院需要填写首次病程

病人转科需要填写转入记录

病人出院需要填写出院记录

医嘱改变：转科医嘱需要填写转科记录

病危医嘱需要填写病危病程记录

手术医嘱需要手术记录

病历改变：有连带关系的病历，如书写了A病历就需要书写B病历，否则

现在可以针对病人表Inpatient、医嘱表、病历记录表RecordDetail表来设置进行触发。

[1]

病人表：根据病人状态Inpatient.Status + “病人状态改变”

医嘱表：根据医嘱代码 + “医嘱改变” \*\*\*\*由于系统没有医嘱录入功能，所以暂不开放\*\*\*\*

病历改变：根据病历类型RecordDetail.MR\_CLASS + “病历改变”

[2]

在QCCondition表中获取Code，然后再通过QCCondition.Code获取指定的QCRule。

**<1>**

如果操作类型是一次性，即QCRule.Mark == 0，在插入前判断原先是否已经插入过记录到QCRecord表中，通过QCRule.Code==QCRecord.RuleCode AND QCRecord.Noofinpat == 当前病人住院号来判断，如果已经存在则不做操作，否则需要插入初始数据到QCRecord表中。

如果操作类型不是一次性，即QCRule.Mark <> 0，则插入初始数据到QCRecord表中。

**<2>**插入的初始数据如下：

如果时限规则QCRule中延迟时间 > 0：

|  |  |
| --- | --- |
| QCRecord.CONDITIONTIME = 触发时间 + 延迟时间 | 条件时间 |
| QCRecord.CONDITION = 0 | 条件未完成 |
| QCRecord.RESULTTime = NULL | 完成时间‘无’ |
| QCRecord.RESULT = 0 | 结果未完成 |
| QCRecord.FOULSTATE = 0 | 未违规 |
| QCRecord.OPERATETIME = SYSDATE | 创建时间 |

如果时限规则QCRule中延迟时间 == 0：

|  |  |
| --- | --- |
| QCRecord.CONDITIONTIME = 触发时间 | 条件时间 |
| QCRecord.CONDITION = 1 | 条件已完成 |
| QCRecord.RESULTTime = NULL | 完成时间‘无’ |
| QCRecord.RESULT = 0 | 结果未完成 |
| QCRecord.FOULSTATE = 0 | 未违规 |
| QCRecord.OPERATETIME = SYSDATE | 创建时间 |

1. **内部定时服务，每隔一段时间判断时限的违规情况。如果针对的是循环时限，需要动态的判断，并插入初始数据到QCRecord表中。**
2. 如果 QCRecord.REALCONDITIONTIME（实际条件时间）>= SYSDATE（当前时间）

则 QCRecord.CONDITION=1，即条件已完成

判断QCRecord表中的记录有无对应的病历，如果没有对应的病历，即QCRecord. RECORDDETAILID == NULL AND QCRecord.CONDITION + QCRecord.TIMELIMIT < SYSDATE

则 QCRecord.FOULSTATE=1，即已违规

1. 时限规则类型是循环

在QCRecord表中遍历Condition==1 AND 病人未出院 AND ISCYCLE==1，通过找到的第一条循环记录以及FirstCycleRecordID==第一条循环记录的ID，来判断是否需要再插入记录到QCRecord表中。

如果判断需要插入记录到QCRecord表中

|  |  |
| --- | --- |
| QCRecord.CONDITIONTIME = 触发时间 | 循环规则中第一条记录的条件时间 + （循环次数 – 1）\*间隔的时间 |
| QCRecord.CONDITION = 1 | 条件已完成 |
| QCRecord.RESULTTime = NULL | 完成时间‘无’ |
| QCRecord.RESULT = 0 | 结果未完成 |
| QCRecord.FOULSTATE = 0 | 未违规 |
| QCRecord.OPERATETIME = SYSDATE | 创建时间 |

1. **结束时限规则时，根据时限的结果判断，更新QCRecord表中的数据，结束时限。**

在书写完病历之后触发，根据NOOFINPAT、QCCODE、病历的创建时间 找到与其对应的QCRECORD表中的记录

如何找到与其对应的QCRECORD表中的那条时限规则记录：

首先在QCRECORD表中判断这个病历是否已经处理过，即RECORDDETAILID == RecordDetail.ID，

如果已经存在，则不做处理直接返回。

如果不存在，则找到该病人所有的时限规则，即NOOFINPAT==Inpatient.NOOFINPAT，通过病历的QCCode在上面找到的时限规则中查找，抓取未完成（result == 0）AND 条件时间Condition最小的记录，这样就找到了那条时限规则记录。

然后更新这条记录。

|  |  |
| --- | --- |
| QCRecord.CONDITION = 1 | 条件已完成 |
| QCRecord.RESULTTime = SYSDATE | 完成时间 |
| QCRecord.RESULT = 1 | 结果已完成 |
| QCRecord.RECORDDETAILID = RECORDDETAIL.ID | 时限规则对应的具体病历 |

如果 RECORDDETAIL.CREATETIME > QCRECORD.CONDITIONTIME + TIMELIMIT

|  |  |
| --- | --- |
| QCRecord.FOULSTATE = 1 | 已违规 |

如果 RECORDDETAIL.CREATETIME <= QCRECORD.CONDITIONTIME + TIMELIMIT

|  |  |
| --- | --- |
| QCRecord.FOULSTATE = 0 | 未违规 |

时限提醒列表中，只显示QCRecord.Conditon == 1（条件已完成）AND QCRecord.RESULT == 0（结果未完成）的记录，如果QCRecord.FOULSTATE = 0（未违规）需要判断 SYSDATE（当前时间）> QCRecord.ConditionTime + QCRecord.TimeLimit，表示记录已经违规，需要显示警告信息，否则显示提示信息即可。

在显示的时候需要根据当前登录人的级别进行筛选，做到向下兼容。

注意区分开医生和护士

**主动方式：**

（1）生成提醒信息，插入初始化数据，如果是循环时限，则只插入循环的第一条记录



（2）针对循环时限的处理，见**被动方式2**

（3）结束提醒，即完成病历后的处理：



# 4.开发计划及工时分配

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **任务名称** | | **计划工作日** | **里程碑** |  |
| 1 | 需求分析 | | 4 | 需求说明书及解决方案 |  |
| 2 | 数据库建模  实体类库 | | 0.5 | 涉及到病历时限的表，以及表对应的实体类 |  |
| 3 | 基  础  数  据  维  护 | 质量控制条件库QCCONDITIOIN  目标效果:增、删、改、查 | 0.5 | 质量控制条件库维护界面完成 |  |
| 4 | 质量控制分类库RULECATEGORY  目标效果:增、删、改、查 | 0.5 | 质量控制规则分类库维护界面完成 |  |
| 5 | 质量控制规则库QCRULE  目标效果:增、删、改、查 | 1 | 质量控制规则库维护界面完成 |  |
| 5 | 质量病历质量控制库EMRQCITEM  目标效果:增、删、改、查 | 0.5 | 质量控制规则库维护界面完成 |  |
| 6 | 内  部  定  时  服  务 | 根据配置信息抓取数据，判断触发时机，生成提醒信息，插入初始化数据 | 2 |  |  |
| 7 | 针对循环时限的规则进行数据的初始化功能，需要注意有限次循环和无限次循环，以及有限次循环的再次触发的问题 | 2 | 循环时限规则初始化数据插入正确 |  |
| 8 | 针对所有未完成的病历时限规则违规的判定 | 1 | 违规病历判断正确 |  |
| 9 | 病历完成之后结束时限提醒 |  | 时限提醒结束后，提醒的信息应该消失 |  |
| 9 | 将上面的功能融合进电子病历的定时服务系统中 | 1 | 定时服务可以正常运行 |  |
| 10 | 针对病历删除后时限提醒再次生效 | | 1 | 生效后的时限提醒应该重新显示出来 |  |
| 12 | 病历时限提醒功能 | | 1 | 正确抓取提醒的记录 |  |
| 13 | 病历提醒功能与现有电子病历系统中需要调用该功能的融合 | | 0.5 | 保证原有系统调用病历提醒功能的正确性 |  |
| 14 | 病历时限评分功能 | | 1 | 完成病历时限评分功能 |  |
| 15 | 测试数据录入 | | 1 |  |  |
| 16 | 系统测试 | | 3 |  |  |

## 5.进度监控

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **任务名称** | | **计划工作日** | **里程碑** |  |
| 1 | 需求分析 | | 4 | 需求说明书及解决方案 | 2013-1-4武文俊 |
| 2 | 数据库建模  实体类库 | | 0.5 | 涉及到病历时限的表，以及表对应的实体类 | 2013-1-5武文俊 |
| 3 | 基  础  数  据  维  护 | 质量控制条件库QCCONDITIOIN  目标效果:增、删、改、查 | 0.5 | 质量控制条件库维护界面完成 | 2013-01-08项令波 |
| 4 | ~~质量控制分类库RULECATEGORY~~  ~~目标效果:增、删、改、查~~ | ~~0.5~~ | ~~质量控制规则分类库维护界面完成~~ | ~~项令波~~ |
| 5 | 质量控制规则库QCRULE  目标效果:增、删、改、查 | 1 | 质量控制规则库维护界面完成 | 2013-01-07项令波 |
| 5 | 质量病历质量控制库EMRQCITEM  目标效果:增、删、改、查 | 0.5 | 质量控制规则库维护界面完成 | 2013-01-09项令波 |
| 6 | 内  部  定  时  服  务 | 根据配置信息抓取数据，判断触发时机，生成提醒信息，插入初始化数据 | 2 |  | 2013-1-7武文俊 |
| 7 | 针对循环时限的规则进行数据的初始化功能，需要注意有限次循环和无限次循环，以及有限次循环的再次触发的问题 | 2 | 循环时限规则初始化数据插入正确 | 2013-1-9武文俊 |
| 8 | 针对所有未完成的病历时限规则违规的判定 | 1 | 违规病历判断正确 | 2013-1-10  武文俊 |
| 9 | 病历完成之后结束时限提醒 | 1 | 时限提醒结束后，提醒的信息应该消失 | 2013-1-11  武文俊 |
| 9 | 将上面的功能融合进电子病历的定时服务系统中 | 1 | 定时服务可以正常运行 | 2013-1-14  武文俊 |
| 10 | 针对病历删除后时限提醒再次生效 | | 1 | 生效后的时限提醒应该重新显示出来 | 2013-1-15  武文俊 |
| 12 | 病历时限提醒功能 | | 1 | 正确抓取提醒的记录 | 2013-01-11  项令波 |
| 13 | 病历提醒功能与现有电子病历系统中需要调用该功能的融合 | | 1 | 保证原有系统调用病历提醒功能的正确性 | 2013-01-14  项令波 |
| 14 | 病历时限评分功能 | | 1 | 完成病历时限评分功能 |  |
| 15 | 测试数据录入 | | 1 |  |  |
| 16 | 系统测试 | | 3 |  |  |

IF foulstate == 0 未违规

IF Sysdate >(realconditiontime + timelimit) --已违规

Sysdate - (realconditiontime + timelimit) 超时的时间 + foulmessage

ELSE –未违规

(realconditiontime + timelimit) – Sysdate 剩余时间 + reminder

IF foulstate ==1 已违规

Sysdate - (realconditiontime + timelimit) 超时的时间 + foulmessage