**YY系统**

**数据库结构设计说明书**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **文档版本修改记录** | | | | | |
| **版本** | **修改日期** | **修改摘要** | **修改人** | **审批人** | **审批日期** |
| V 1.0 | 2019-1-23 | 创建 | 张三 | 李四 | 2019-1-23 |
|  |  |  |  |  |  |

**ZZZZ有限公司**

**2019年1月**

**目录**

[第1章 引言 3](#_Toc32187)

[1.1. 目的 3](#_Toc7059)

[1.2. 背景 3](#_Toc8272)

[第2章 数据库结构设计 4](#_Toc22590)

[2.1. 数据库分布 4](#_Toc31540)

[2.2. 命名规则 4](#_Toc6786)

[2.3. 物理结构设计 10](#_Toc10931)

[第3章 数据存储设计 13](#_Toc23491)

[第4章 数据采集 14](#_Toc24543)

[4.1. 数据来源和范围 14](#_Toc6421)

[4.2. 数据采集方法 14](#_Toc14977)

[4.3. 数据输入介质和设备 14](#_Toc22021)

# 引言

## 目的

编写数据库设计说明书的目的是：详细定义数据库的表名、字段名、约束条件等数据信息；明确数据处理流程、表之间的逻辑关系，以便开发人员进行项目设计，同时也为后期的数据库维护工作提供良好的使用说明。

本文档的预期的读者包括建设方的项目管理人员、业务需求人员、质量控制人员、监理方、软件评测方；承建方的管理、设计、开发、测试人员以及在系统的设计、开发、装配、测试、验收、维护等过程中可能关注数据库设计信息的人员。

## 背景

居民需要的XXXX涉及多项具体的业务流程，包括诊前的咨询、导诊、预约，诊间的候诊、支付、查询，以及诊后的评价、随访等。本项目针对XXXX应用，将通过与XX平台、38家市级医院现有应用、服务和数据进行对接整合，将这些服务从多个业务流程中封装出来，整合在一起，为居民提供统一的服务入口，方便访问操作。

医院预检是病人进入医院就医的第一个环节,预检的效率直接关系到医院整个业务运作状况。基于Web服务组合的预检系统从提高预检的准确性出发,简化了工作环节,提高了工作效率,同时又可以方便病人就医,改善医院的就医环境,进而强化医院的自身管理,节药医院的管理费用开支和成本。在已有平台基础上应用服务组合的方式来实现医院预检系统的应用,以达到优化就诊环境、减轻医护人员工作量的目的。

# 数据库结构设计

## 数据库分布

本系统使用Mysql作为数据库存储系统。

## 命名规则

### 表命名

必须为表名加功能分类。

命名：[分类名]\_[表友好名]（省略前缀：[数据库名简写]\_TBL\_）

范例：TB\_JBGL\_MZFGLK（接口表\_疾病管理慢阻肺管理卡）

**TB：接口表**

JBGL\_MZFGLK：疾病管理慢阻肺管理卡（表友好名）

**固定表分类**

TB\_: 接口表(除字典)，如TB\_JBGL\_MZFGLK疾病管理慢阻肺管理卡

TB\_DIC\_：字典表类，如TB\_DIC\_GENDER性别字典表

SYS\_：系统信息类，如SYS\_CHECK\_RESULT检验结果日志表

FZ\_：配置辅助类，如FZ\_DEF\_TABLE接口表定义

BUS\_：业务类，注意策略中的要求，如果表达到100W以上要用分区

### 表分区

p ，分区名必须有特定含义的单词或字串。例如 :TB\_JBGL\_MZFGLK 的分区p2004100101表示该分区存储 2004100101时段的数据。

字段：必须以有特征含义的单词或缩写组成，中间可以用“\_”分割。例如：USER\_NAME。

主键：PK\_。主键名称应是 前缀+表名+构成的字段名。如果复合主键的构成字段较多，则只包含第一个字段。表名可以去掉前缀。

索引：IX\_。索引名称应是 前缀+表名+构成的字段名。如果复合索引的构成字段较多，则只包含第一个字段，并添加序号。表名可以去掉前缀。

外键：FK\_。外键名称应是 前缀+ 外键表名 + 主键表名 + 外键表构成的字段名。表名可以去掉前缀。

视图：V\_.按业务操作命名视图。

实体化视图：MV\_。按业务操作命名实体化视图。

存储过程：P\_ 。按业务操作命名存储过程

触发器：Trg\_ 。触发器名应是 前缀 + 表名 + 触发器名。

函数：F\_ 。按业务操作命名函数

包：Pkg\_ 。按业务操作集合命名包。

序列：Seq\_ 。按业务属性命名。

表空间：公用表空间，Tbs\_ 。 根据存储的特性命名，例如： tbs\_parameter 。

专用表空间：Tbs\_表名称\_nn。该表空间存储特定表，或表分区的数据

数据文件：表空间nn.dbf ，nn =1，2，3，4，…

普通变量：Var\_ 。

游标变量：Cur\_ 。存放游标记录集。

记录型变量：Rec\_ 。 存放记录型数据。

表类型变量：Tab\_ 。 存放表类型数据。

### 一般规范

语言：命名使用英文单词，不使用复数。英文单词使用同对象本身意义相对或相近的单词。选择最简单或最通用的单词。不能使用毫不相干的单词来命名。当一个单词不能表达对象含义时，用词组组合，如果组合太长时，采用简写或缩写，缩写要基本能表达原单词的意义。当出现对象名重名时，是不同类型对象时，加类型前缀或后缀以示区别。

大小写：名称一律大写，以方便不同数据库移植，以及避免程序调用问题，单词分隔，命名的各单词之间使用下划线“\_”进行分隔，命名的各单词之间不允许有空格存在。

保留字：命名不允许使用SQL保留字

命名长度：表名、字段名、视图名长度应限制在30个字符内(含前缀)

字段名称：同一个字段名在一个数据库中只能代表一个意思，不同的表用于相同内容的字段应该采用同样的名称，字段类型定义。

### 数据类型

字符型：固定长度的字串类型采用char，长度不固定的字串类型采用varchar2避免在长度不固定的情况下采用char类型。如果在数据迁移等出现以上情况，则必须使用trim()函数截去字串后的空格。

数字型：数字型字段尽量采用number类型。

日期和时间：系统时间，由数据库产生的系统时间首选数据库的日期型，如DATE类型。

外部时间：由数据导入或外部应用程序产生的日期时间类型采用varchar2类型，数据格式采用：YYYYMMDDHH24MISS。

大字段：如无特别需要，避免使用大字段(blob，clob，long，text，image等)。

唯一键：对于数字型唯一键值，尽可能用系列sequence产生。

### 设计

范式：如无性能上的必须原因，应该使用关系数据库理论，达到较高的范式，避免数据冗余,但是如果在数据量上与性能上无特别要求，考虑到实现的方便性可以有适当的数据冗余，但基本上要达到3NF（要求一个数据库表中不包含已在其它表中已包含的非主关键字信息）。如非确实必要，避免一个字段中存储多个标志的做法。如11101表示5个标志的一种取值。这往往是增加复杂度，降低性能的地方。

表设计：每个表必须包含以下属性参数: TABLESPACE，PCTUSED，STORAGE。根据表的不同属性进行表类型设计：

分区表：对于数据量比较大的表，根据表数据的属性进行分区，以得到较好的性能：如果表按某些字段进行增长，则采用按字段值范围进行范围分区。如果表按某个字段的几个关键值进行分布，则采用列表分区。对于静态表，则采用hash分区或列表分区。在范围分区中，如果数据按某关键字段均衡分布，则采用子分区的复合分区方法。

聚蔟表：如果某几个静态表关系比较密切，则可以采用聚蔟表的方法。

索引：对于查询中需要作为查询条件的字段，可以考虑建立索引。最终根据性能的需要决定是否建立索引。对于复合索引，索引字段顺序比较关键，把查询频率比较高的字段排在索引组合的最前面。在分区表中，尽量采用local分区索引以方便分区维护。索引的创建要与应用结合考虑，建议大的OLTP表不要超过6个索引。 尽可能的使用索引字段作为查询条件，尤其是聚簇索引，必要时可以通过index index\_name来强制指定索引。避免对大表查询时进行table scan，必要时考虑新建索引。在使用索引字段作为条件时，如果该索引是联合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用。 要注意索引的维护，周期性重建索引，重新编译存储过程。

主键：关联表的父表要求有主健。

外键：对于关联两个表的字段，一般应该分别建立主键、外键。实际是否建立外键，根据对数据完整性的要求决定。为了提高性能，要求对外健建立索引。

tempdb的使用规范：尽量避免使用distinct、order by、group by、having、join，因为这些语句会加重tempdb的负担。 避免频繁创建和删除临时表，减少系统表资源的消耗。 在新建临时表时，如果一次性插入数据量很大，那么可以使用select into代替create table，避免log，提高速度；如果数据量不大，为了缓和系统表的资源，建议先create table，然后insert。 如果临时表的数据量较大，需要建立索引，那么应该将创建临时表和建立索引的过程放在单独一个子存储过程中，这样才能保证系统能够很好的使用到该临时表的索引。 如果使用到了临时表，最后务必将所有的临时表显式删除，先truncate table，然后drop table，这样可以避免系统表的较长时间锁定。 慎用大的临时表与其他大表的连接查询和修改，减低系统表负担，因为这种操作会在一条语句中多次使用tempdb的系统表。

NULL值：对于字段能否null，应该在sql建表脚本中明确指明，不应使用缺省。由于NULL值在参加任何运算中，结果均为NULL。所以在应用程序中必须利用nvl()函数把可能为NULL值得字段或变量转换为非NULL的默认值。例如：NVL（sale,0）。

注释：表、字段等应该有中文名称注释，以及需要说明的内容。

### SQL编写

* 字符类型数据

SQL中的字符类型数据应该统一使用单引号。特别对纯数字的字串，必须用单引号，否则会导致内部转换而引起性能问题或索引失效问题。利用trim(),lower()等函数格式化匹配条件。

* 复杂SQL

对于非常复杂的SQL(特别是有多层嵌套，带子句或相关查询的)，应该先考虑是否设计不当引起的。对于一些复杂SQL可以考虑使用程序实现。

* 高效性
* 避免In子句

使用In 或 not In子句时，特别是当子句中有多个值时，且查询数据表数据较多时，速度会明显下降。可以采用连接查询或外连接查询来提高性能。

* 避免嵌套的Select子句

这个实际上是In子句的特例。

* 避免使用Select \* 语句

如果不是必要取出所有数据，不要用\*来代替，应给出字段列表。

* 避免不必要的排序

不必要的数据排序大大的降低系统性能。

* 健壮性
* Insert语句

使用Insert语句一定要给出要插入值的字段列表，这样即使更改了表结构加了字段也不会影响现有系统的运行。

* 安全性
* Where 条件

无论在使用Select,还是使用破坏力极大的Update和Delete语句时，一定要检查Where条件判断的完整性，不要在运行时出现数据的重大丢失.如果不确定，最好先用Select语句带上相同条件来搜一下结果集，来检验条件是否正确.

* 其他

1. 尽量避免大事务操作，慎用holdlock子句，提高系统并发能力。
2. 尽量避免反复访问同一张或几张表，尤其是数据量较大的表，可以考虑先根据条件提取数据到临时表中，然后再做连接。
3. 尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该改写；如果使用了游标，就要尽量避免在游标循环中再进行表连接的操作。
4. 注意where字句写法，必须考虑语句顺序，应该根据索引顺序、范围大小来确定条件子句的前后顺序，尽可能的让字段顺序与索引顺序相一致，范围从大到小。
5. 不要在where子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。
6. 尽量使用exists代替select count(1)来判断是否存在记录，count函数只有在统计表中所有行数时使用，而且count(1)比count(\*)更有效率。
7. 尽量使用“>=”，不要使用“>”。
8. 注意一些or子句和union子句之间的替换
9. 注意表之间连接的数据类型，避免不同类型数据之间的连接。
10. 注意存储过程中参数和数据类型的关系。

注意insert、update操作的数据量，防止与其他应用冲突。如果数据量超过200个数据页面（400k），那么系统将会进行锁升级，页级锁会升级成表级锁。

## 物理结构设计

# 数据存储设计

本程序在高级语言JAVA进行编码，直接的内存分配由JAVA运行时分配。

本组件内所依赖的变量、结构要求全部在组件元素内申明。

# 数据采集

## 数据来源和范围

用户输入数据和各家医院端接口调用数据。

## 数据采集方法

应用系统采集，以及其他库调用数据。

## 数据输入介质和设备

用户手机应用。