**Oracle数据库设计规范**

# 1 数据库对象命名规范

## 1.1通用规范

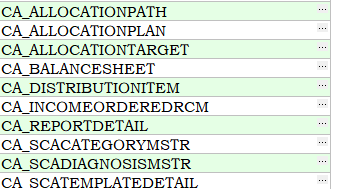
* **使用英文**：要用简单明了的英文单词，不要用拼音，特别是拼音缩写。主要目的是让人容易明白这个对象是做什么用的；
* **一律大写，特别是表名**：有些数据库，表的命名乃至其他数据对象的命名是大小写敏感的，为了避免不必要的麻烦，并且尊重通常的习惯，最好一律用大写；
* **尽量不要用简写**：不要怕表名或者字段名长，除非是超过ORACLE的限制才使用简写

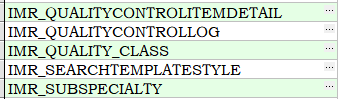
## 1.2 表的命名

* 使用与该表关联比较密切的表类似的单词命名；

例如：费用明细相关系列的表：PATIENTACCOUNT（费用明细头表）、PATIENTACCOUNTTXN（费用明细表）、PATIENTACCOUNTTXNHISTORY（费用明细变化历史表）、PATIENTACCOUNTTXNDISCOUNT费用明细折扣变化历史表）；

* **为了不与其他项目或者其他系统、子系统的表重名**；
* **表示某种从属关系**：例如表明是属于某个子系统、某个模块或者某个项目等等。表示这种从属关系的一个主要目的是，从表名能够大概知道如何去找相关的模块、相关的人员。
* 独立子系统可以设置前缀，例如CA\_（数字签名）,CHI\_（商业保险）

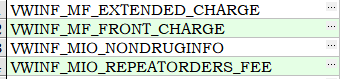




* 一些固定模式命名：
* 明细表：\*\*\*DETAIL,例如：BILLDETAIL（账单明细表）
* 历史表：\*\*\*HISTORY,例如：RECEIPTHISTORY(发票历史表)
* 策略表/策略明细表：\*\*\*POLICY, \*\*\*POLICYDETAIL,例如：OMSURCHARGEPOLICY（附加收费策略表）、OMSURCHARGEPOLICYDETAIL（医嘱附件收费策略表）
* 交易表：\*\*\*TXN，例如：MATERIALTXN（物品交易表）
* 字典表：\*\*\*MSTR，例如：USERMSTR（用户字典表）
* 结果表：\*\*\*SULT，例如：LISRESULT（检验结果表）
* LOG表：\*\*\*LOG，例如：EHEALTHTASKEXECLOG（定时任务执行log表）

## 视图命名

VWINF\_模块/系统简称\_具体用途；例如VWINF\_CDR\_PATIENTINFO



## 1.4 程序包命名

PG程序包名：例如PGOM,PGPA，PGADT

注意：项目组可能自行修改的内容千万不能放到PG程序包中,而是独立做成函数或者存储过程，因为家里可能会升级覆盖现场的包

## 1.5 存储过程命名

pr模块简称\_具体用途：例如prSms\_GetContent

## 1.6 函数命名

fx模块简称\_具体用途:例如fxEmr\_IsWriteRight,fxSys\_ReplaceClob

## 1.7 触发器命名

* 更新后触发：TG\_AF\_IU\_表名/具体用途 ,例如： TG\_AF\_IU\_VISIT\_EVE，TG\_AF\_U\_MATERIALITEMMSTR
* 更新前触发：TG\_BF\_IU\_表名/具体用途，例如： TG\_BF\_IU\_VISIT

注意：

* 很少用到更新前触发，除非是要做一些拦截操作；
* I表示插入，U表示更新，如果只需要扑获其中一种，则命名中只需要写对应的

## 1.8主键索引命名

PK\_表名；

例如：PATIENTACCOUNTTXNDISCOUNT表的主键索引：PK\_PATIENTACCOUNTTXNDISCOUNT

注意：所有表的表都必须要有主键索引

## 1.9 外键命名

FK\_本表表名缩写\_关联表表名缩写；

例如：

PATIENTACCOUNTTXNDISCOUNT表的PATIENTACCOUNTTXN\_ID字段的外键：FK\_PATD\_PAT

例如：BEDHISTORY表的BEDMSTR\_ID字段的外键：FK\_BH\_BM

注意：外键一定要建索引

## 1.11 索引命名

NDX\_表名缩写\_字段名/字段名缩写；

例如：PATIENTACCOUNTTXNDISCOUNT表的PATIENTACCOUNTTXN\_ID字段的索引：NDX\_PATD\_PAT\_ID

例如：BEDHISTORY表的BEDMSTR\_ID字段的索引：NDX\_BH\_BEDMSTR\_ID

# 2 数据库对象设计原则

## 2.1 表的设计

### 2.1.1 主、外键

(1) 每个表，都必须要有主键（NUMBER(38)）。主键是每行数据的唯一标识，保证主键不可随意更新修改，它会为你的程序以及将来查找数据中的错误等等，提供一定的帮助；

(2) 一个表的某列与另一表有关联关系的时候，如果加得上的话，请加上外键约束。外键是很重要的，所以要特别强调：

* 主从表建外键，没有主从关系的不叫外键约束，但加索引；不要以数据操作不方便为理由而不建外键。是的，加上外键以后，一些数据操作变得有些麻烦，但是这正是对数据一致性的保护。正是因为这种保护很有效，所以最好不要拒绝它；

（3）两表的关联关系一般放在下游表，减少回更新ID；上游表可适当做下游表的字段冗余；

### 2.1.2 列的设计

(1) 字段的宽度要在一定时间内足够用，但也不要过宽，占用过多的存储空间,对于长度不确定的列，采用可变长度的数据类型如 varchar类型；

（2）字段的类型及宽度在设计以及后面进行开发时，往往要与应用的设计、开发人员商讨，以得到双方认可的类型及宽度；

（3）除非必要，否则尽量不加冗余列。所谓冗余列，是指能通过其他列计算出来的列，或者是与某列表达同一含义的列，或者是从其他表复制过来的列等等。冗余列需要应用程序来维护一致性，相关列的值改变的时候，冗余列也需要随之修改，而这一规则未必所有人都知道，就有可能因此发生不一致的情况。如果是应用的特殊需要，或者是为了优化某些逻辑很复杂的查询等操作，可以加冗余列；

（4）除非必要，否则尽量不使用LONG, TEXT, BLOB, CLOB, NCLOB, LONG, LONG RAW这一类的数据类型，而是使用其他可以替代的数据类型；优先使用varchar2类型替代CHAR类型，除非列宽有严格的要求而且得到应用严格支持；

（5）允许为空的列，代码一定要处理该字段为空的情况；

（6）IND字段一定不能为空，并且需要设置默认值；

（7）描述业务流水的表一般没有逻辑删除字段；

（8）对已有业务进行功能升级时，如果数据结构发生变化，一定要做好数据迁移、新老业务衔接工作，有重大变化时需发邮件通知项目组；

（9）业务流程设计完成后，需要从数据统计分析需求方面看表结构的设计是否合理；

### 2.1.3 记录数

（1）单表的记录数一般控制在两千万条 (参考值，各应用可以根据实际情况进行适量调整) 以内；

（2）记录数在两千万和两亿条之间的表一定要采用分区技术，并根据应用的使用情况创建合适的分区标准，单个分区内的记录数一般控制在两千万条(参考值，各应用可以根据实际情况进行适量调整)以内，同时表的索引使用对应的分区索引；

（3）记录数超过五千万条的表一定要考虑信息生命周期，必须考虑历史数据的剥离，并在应用设计中完成对历史数据的相应处理功能（历史数据的剥离规则须经业务使用部门的确认）；

### 2.2 索引的设计

索引是从数据库中获取数据的最高效方式之一。95%的数据库性能问题都可以采用索引技术得到解决。但大量的DML操作（增删改查）会增加系统对索引的维护成本，对性能会有一定影响，对于插入相当频繁的表要慎重建索引，索引也会占相当的存储空间，所以要根据硬件环境和应用需求在空间和时间上达到最好的平衡点，主要原则：

* 适当利用索引提高查询速度：当数据量比较大，了解应用程序的会有哪些查询，依据这些查询需求建相应的索引；最好亲自试验一下，模拟一下生产环境的数据量，在此数据量下，比较一下建索引前后的查询速度；索引对性能会有一定影响，对于DML频繁列的索引要定期维护（重建）。但是，索引的结构对于索引的更新（例如在插入数据的时候）是有一定优化的，所以不要在没有试验以前过分夸大它对性能的影响。最终还是以试验为准；
* 不要建实际用不上的索引，与上条相关，如果建的索引并不提高任何一应用中的查询速度，则要把它删除；有些数据库有相关工具可以发现实际未被使用的索引，可以利用一下；
* 索引类型的选择：

要根据数据分布及应用来决定如何建立索引，一般的高基数数据列（高基数数据列是指该列有很多不同的值）时 ,建立BTree索引（一般数据库索引的缺省类型）；当低基数数据列（该列有大量相同的值）时，可以考虑建立位图索引（如果所选数据库支持的话），但位图索引是压缩类型索引，所以DML（增、删、改）的代价更高，要综合考虑；

* 索引列的选择：

如果检索条件有可能包含多列，创建联合主键或者联合索引，把最常用于检索条件的列放在最前端，其他的列排在后面；不要索引使用频繁的小型表，假如这些小表有频繁的DML就更不要建立索引，维护索引的代价远远高于扫描表的代价；

* 主键索引在建立的时候一定要明确的指定名称，不能让系统默认建立主键索引（可能有些数据库无法指定主键名，则例外）；
* 外键必须需建索引。当有一定数据量，并且经常以外键所在列为关联，进行关联查询时，需要建索引（可能有些数据库自动为外键建索引，则例外）；
* 控制一个表的索引数量，尽量使得一个表的索引数量小于五个；

## 2.3 视图的设计

（1）在不太清楚视图用法的情况下，尽量不建。因为一旦建了，就有被滥用的危险；

（2）如果需要建视图，只要是打算长期使用的，请写入数据库设计中。明确它的用途、目的；

（3）建立视图时要明确写出所有要选择出的列名而不要以**SELECT \***来代替，可以使结构清晰可读性增强，也不会增加它对表的所有字段的依赖，而表是很可能修改的，特别是增加字段。就很有可能导致使用该视图的应用程序出错；

## 2.4 存储过程、函数、触发器的设计

存储过程、函数仅维护现有的，不允许新建新的存储过程、函数，禁用触发器！。使用存储过程、函数、触发器的影响：

* 增强了应用对数据库的依赖，如果打算将来移植数据库的话，使用得越多，则移植的困难越大；数据库中的业务逻辑越多（存储过程等），应用以及存储过程等的维护难度也会增大；
* 通常存储过程等没有面向对象的特性，不容易设计出易于扩展的结构。当存储过程比较复杂时，或者它们相互间的调用关系比较复杂时，可能难于维护；

# 3 SQL的设计和使用

## 3.1 Sql 书写规范

（1）尽量不要写复杂的SQL：过于复杂的S**QL**可以用存储过程或函数来代替，效率更高；甚至如果能保证不造成瓶颈的话，把条**SQL**拆成多条也是可以的。这与一般的编码规范很相似的，首先是要易懂。易懂也就意味着容易维护，对较为复杂的**sql**语句加上注释，说明算法、功能注释风格：注释单独成行、放在语句前面。

（2）应对不易理解的分支条件表达式加注释；

（3）对重要的计算应说明其功能；

（4）过长的函数实现，应将其语句按实现的功能分段加以概括性说明；；

（6）常量及变量命名要使用英文命名，取该常量、变量的取值含义进行命名，无法准确描述时适当给予注释；

（7）尽量采用多行注释，避免在维护过程中因格式调整导致的部分代码失效；

（8）连接符or、in、and、以及＝、<=、>=等前后加上一个空格，便于阅读；

（9）不要用SELECT \*：

* SELECT语句中写出必要的要选择的全部列名，增强语句可读性，避免不必要的选择；
* SELECT \* 增加了对所有字段的依赖，当表增加了字段后，有可能发生错误；
* 此外还可能增加了数据的流量，查询了一些实际不需要的字段；

（10）避免长事务（Transaction）：

长事务容易造成死锁，应该避免，单个事务使用的数据库和系统资源不宜超过总资源1-2%（参考值，各应用可以根据实际情况进行适量调整,这种情况不适用于数据仓库）；

（11）行最长不能超过80字符,同一语句不同字句之间逗号以后空格,其他分割符前空格 where子句书写时，每个条件占一行，语句另起一行时，以保留字或者连接符开始，连接符右对齐；

（12）多表连接时，使用表的别名来引用列；

（13）SQL中对视图的引用：在不太清楚视图用法的情况下，尽量不用。只是因为视图中有自己想要的字段就拿来用，是相当普遍和错误的用法。原因如下：

* 增加了不必要的数据流量，对你的实际需求，那很可能是一个非常复杂的视图，有大量你不需要的字段，并且关联了很多你实际不需要的表，对数据库资源会有过多的消耗；
* 增加了应用程序对视图的依赖，不必要的依赖是越少越好的。当有应用程序依赖了某个视图，不久可能其他人因为某种原因会修改此视图，原来的应用有可能会受到不同程度的影响；

## 3.2 *SQL 性能优化建议*

*（1）系统可能选择基于规则的优化器，所以将结果集返回数据量小的表作为驱动表，即将结果集返回数据量小的表放在FROM后边最后一个表；*

*（2）大量的排序操作影响系统性能，所以尽量减少order by和group by排序操作；*

*（3）如必须使用排序操作，排序尽量建立在有索引的列上；*

*（4）索引的使用*

* *尽量避免对索引列进行计算。如对索引列计算较多，请提请数据库管理员建立函数索引；*
* *尽量注意比较值与索引列数据类型的一致性(number与number比较、char与char比较)，避免使用数据库的类型自动转换功能；如：SELECT \* FROM category WHERE id = ‘123’; -- id’s type is number*
* *对于复合索引，SQL语句必须使用主索引列；*
* *索引字段中，尽量避免使用NULL值；*
* *对于索引的比较，尽量避免使用NOT=（!=）*
* *查询列和排序列与索引列次序保持一致 ；*

*（5）查询的WHERE过滤原则，应使过滤记录数最多的条件放在最前面；*

*（6）in、or子句常会使索引失效，尽可能不使用in、or；*

*（7）尽量避免在循环中使用SQL语句，或在循环中尽量使用动态SQL语句提高执行性能；*

# 4 脚本命名规范和SVN的存放路径

## 4.1 脚本命名规范

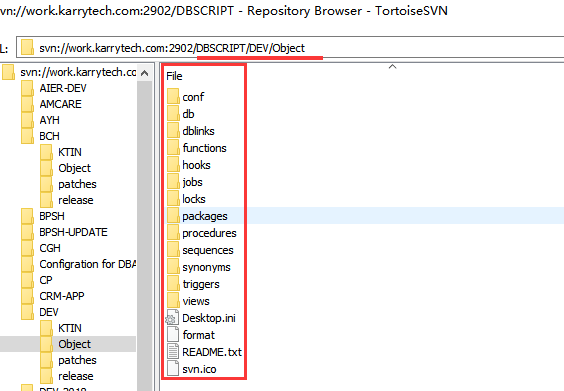
整体命名规则：日期\_操作\_Table\_表名\_操作\_字段名/参数名，其中“日期”形式为“年-月-日”，“操作”首字母大写，“表名”全部大写，“字段名”全部大写。

* 创建一个新表：日期\_ Create\_ Table\_表名；例如：2019-10-12\_Create\_Table\_PACKAGETXNCOUNT
* 增加一个、两个新字段：日期\_Alter\_ Table\_表名\_Add\_字段名；例如：2019-10-12\_Alter\_Table\_REALPACKAGEMSTR\_Add\_PART\_PAY\_IND
* 增加三个及三个以上字段时，在字段名后加\_MORE，能够从文件名中看出此次新增了多个字段
* 增加一个新参数：日期\_Insert\_表名\_参数名；例如：2019-06-26\_Insert\_PARAMETER\_IP\_ORDERSTATION\_SHOW\_PAGETURN
* 新增codemstr表时，脚本名称中的操作应写insert；
* 给已有codemstr加值时，脚本名称中的操作应写update；
* 业务数据迁移或者补历史数据，新生成数据用insert，修改老的数据用update；
* 程序包、存储过程、视图、函数方法可不加日期

## 4.2 SVN的存放规则

（1）先找到对应的项目；

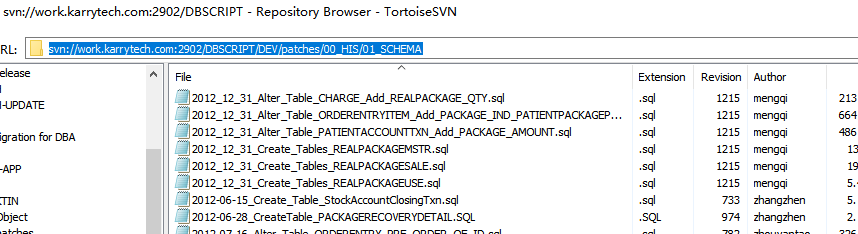
（2）程序包、视图、触发器、函数、存储过程，放在项目下的“Object”路径下，找到对应的类型；



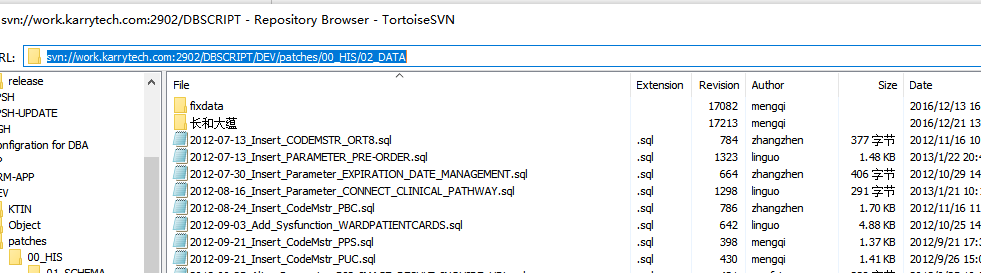
（3）关于表修改的脚本放在项目下的“patches”路径下；

（4）找到“patches”路径下对应的项目，HIS的脚本通常放在“00\_HIS”下；

（5）“00\_HIS”下表结构相关的脚本放在“01\_SCHEMA”路径下；



（6）“00\_HIS”下表结构相关的脚本放在“02\_DATA”路径下；

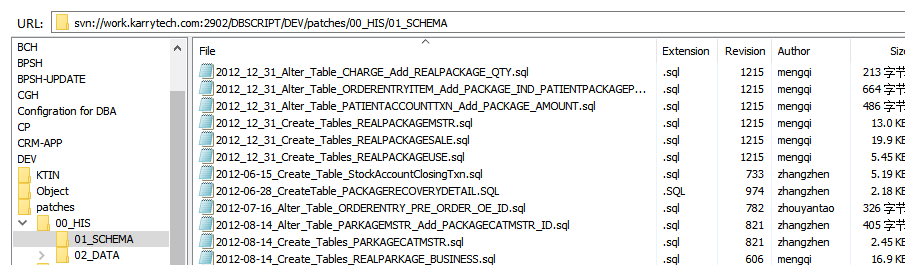


## 4.2 脚本的存放及操作步骤实例（系统参数、页面参数、codemstr）

### 4.2.1 Dev脚本处理

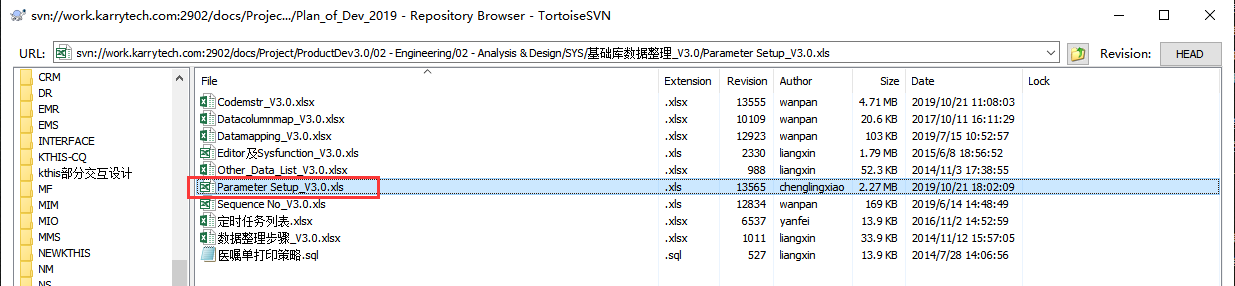
#### 4.2.1.1 dev中某表需要增加字段

（1）dev某表需求增加字段，需存放路径为：

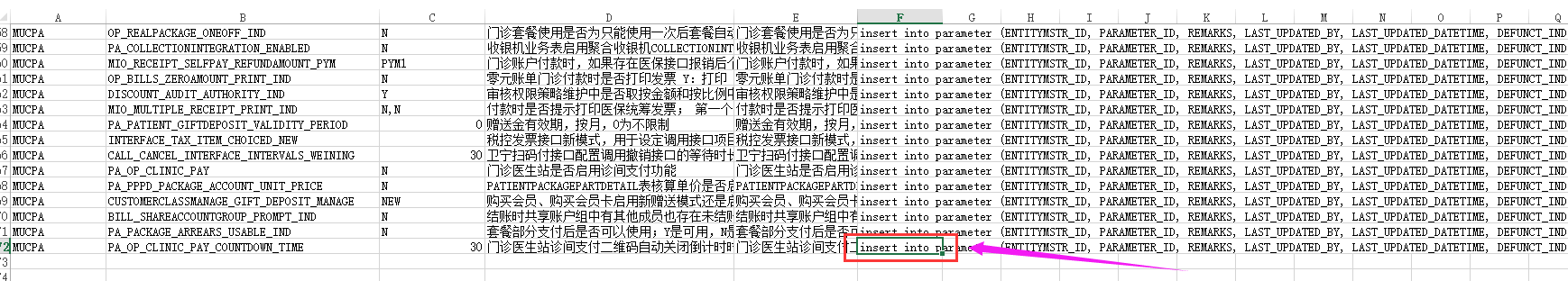


#### 4.2.1.2 dev中需要增加某系统参数

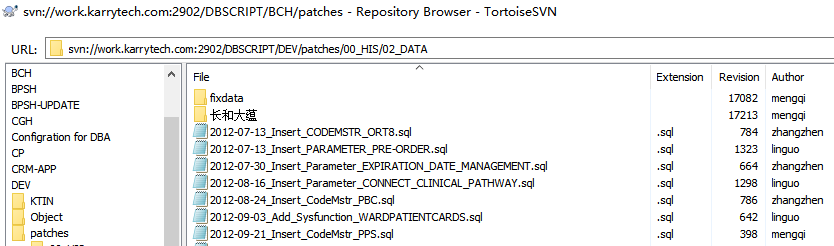
（1）在系统参数表格中找到对应的模块赠加该系统参数，路径为：



（2）拷贝出已生成对应sql，并提交修改内容，注意对比资源，不要冲突；

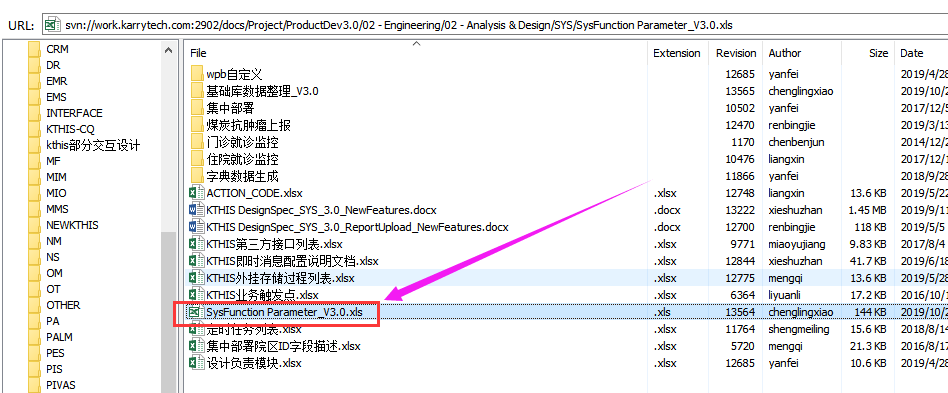


（3）制作成脚本后上传到SVN，路径为：

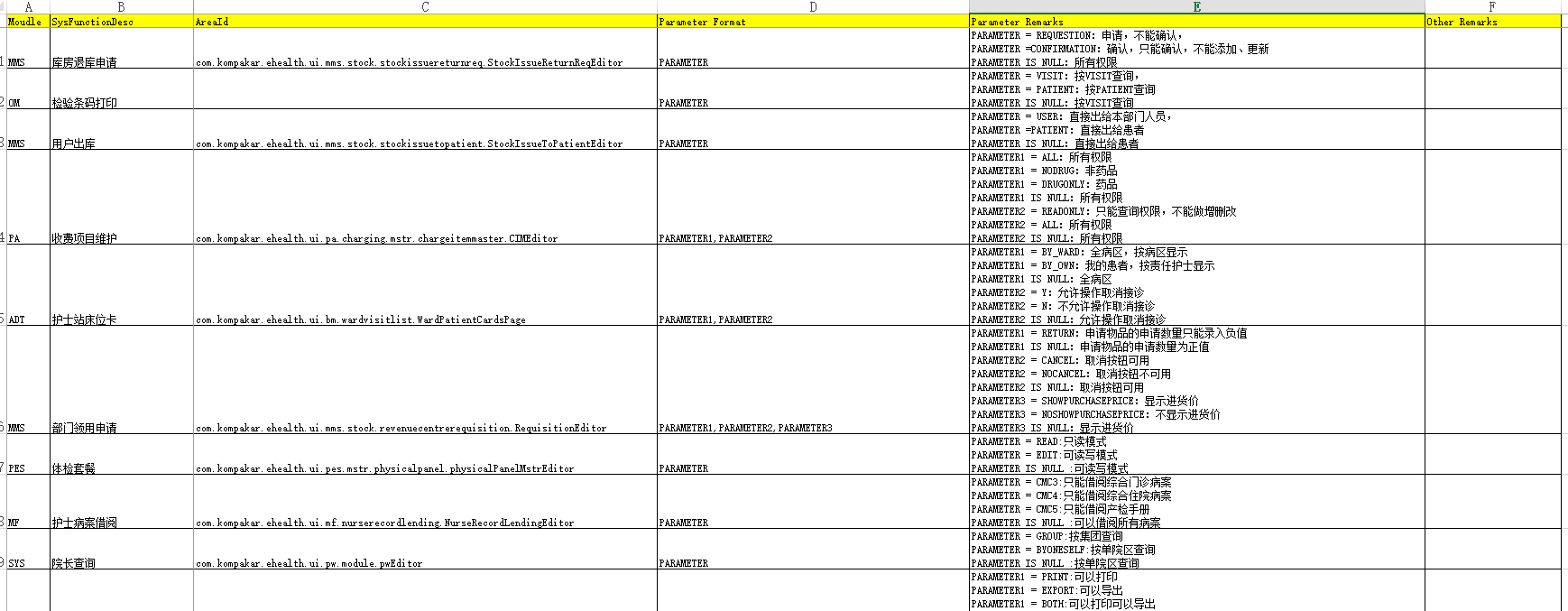


#### 4.2.1.3 dev中需要增加页面参数

（1）在页面参数表格中找到对应的模块赠加该页面参数，路径为：



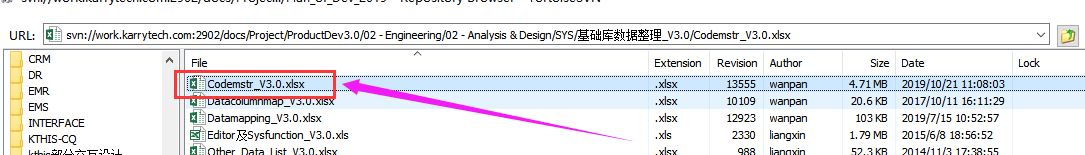
（2）找到对应的页面进行更新，并提交修改内容，注意对比资源，不要冲突；



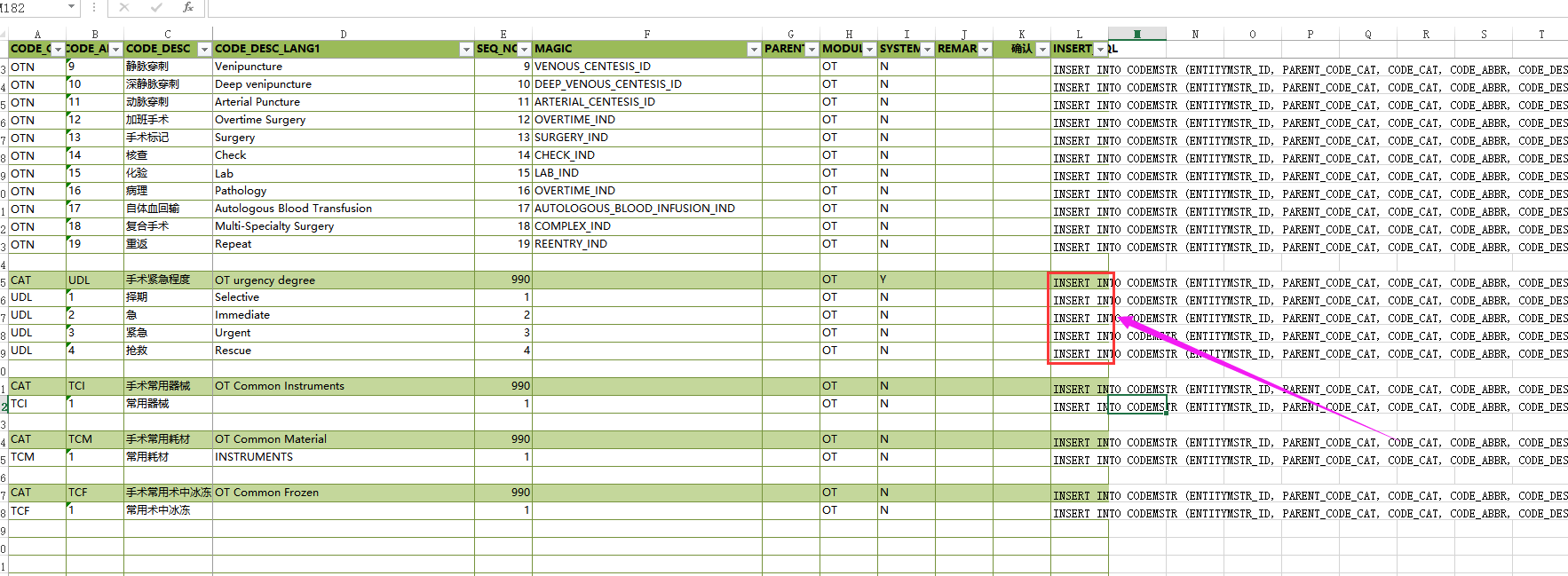
（3）不需要制作对应脚本，需在设计文档中写明；

#### 4.2.1.4 dev中需求增加某codemstr

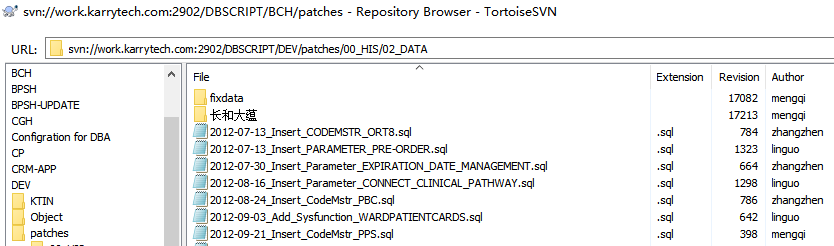
（1）在codemstr表格中找到对应的模块赠加该codemstr，路径为：



（2）拷贝出已生成对应sql，并提交修改内容，注意对比资源，不要冲突；

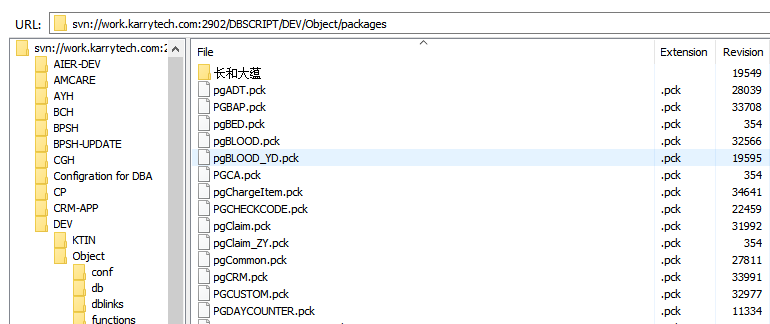


（3）制作成脚本后上传到SVN，路径为：



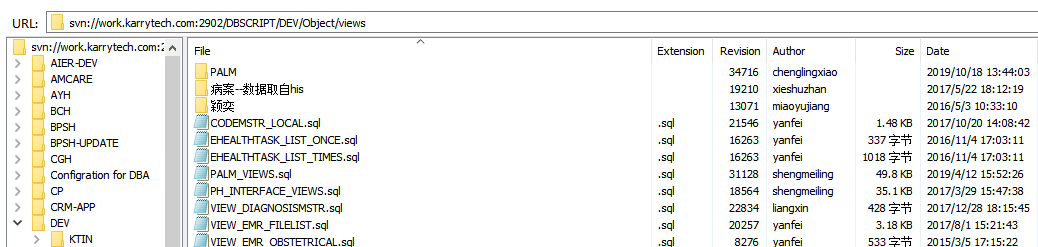
#### 4.2.1.5 dev中修改/新增某程序包

（1）上传的路径为：



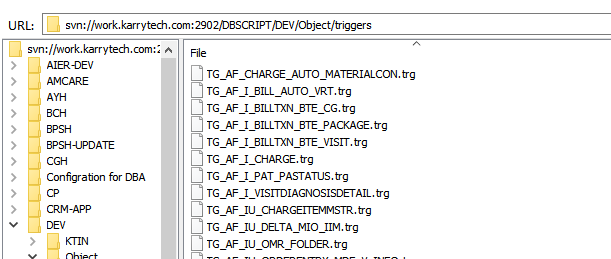
#### 4.2.1.6 dev中修改/新增某视图

（1）上传的路径为：



#### 4.2.1.7 dev中修改/新增某触发器

（1）上传的路径为：



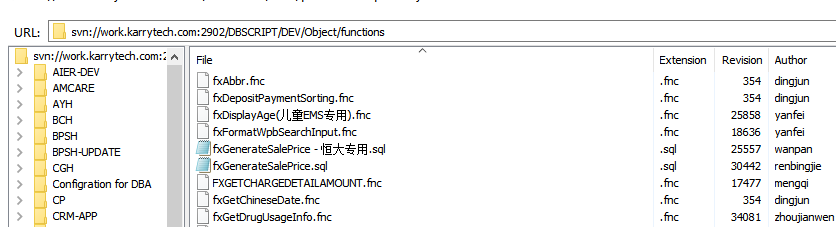
#### 4.2.1.8 dev中修改/新增某存储过程

（1）上传的路径为：



#### 4.2.1.9 dev中修改/新增某函数方法

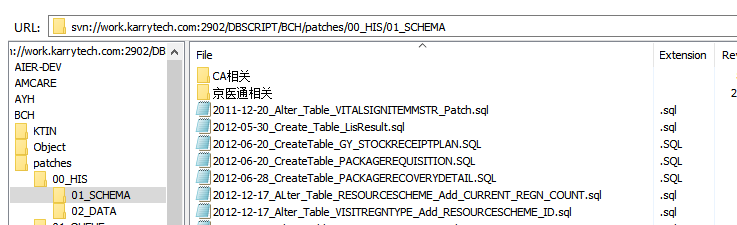
（1）上传的路径为：



### 4.2.2 非Dev脚本处理

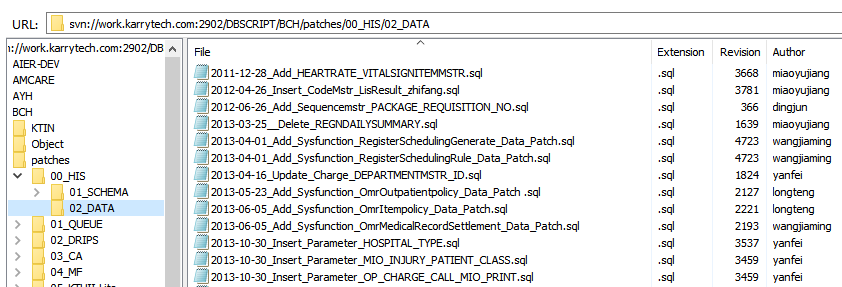
#### 4.2.2.1 胸科项目增加某表或某字段

（1）制作脚本，并上传至：

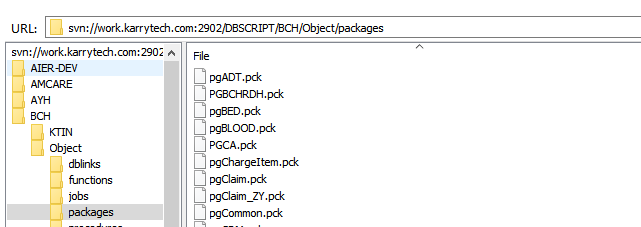


#### 4.2.2.2 胸科项目增加系统参数/codemstr

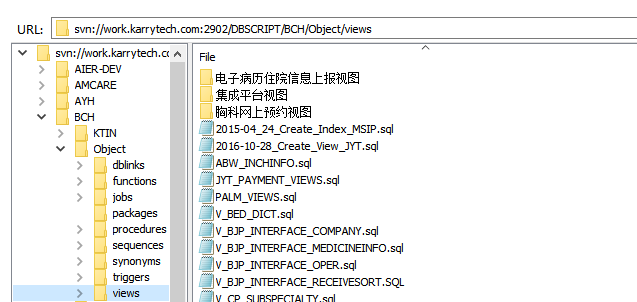
（1）制作脚本，并上传至：



#### 4.2.2.3 胸科项目修改/新增程序包：



#### 4.2.2.4 胸科项目修改/新增视图：



# 5 his中常用字段命名、类型及长度

## 5.1 his中固定字段的命名

|  |  |  |
| --- | --- | --- |
| **字段** | **命名** | **长度** |
| 生效日期 | EFFECTIVE\_DATE | DATE |
| 失效日期 | EXPIRY\_DATE | DATE |
| 生效时间 | EFFECTIVE\_DATETIME | DATE |
| 失效时间 | EXPIRY\_DATETIME | DATE |
| 有效期 | EXPIRE\_DATE | DATE |
| 创建人 | CREATED\_BY | NUMBER(38) |
| 创建时间 | CREATED\_DATETIME | DATE |
| 更新人 | LAST\_UPDATED\_BY | NUMBER(38) |
| 更新时间 | LAST\_UPDATED\_DATETIME | DATE |
| 操作人 | ENTERED\_BY | NUMBER(38) |
| 操作时间 | ENTERED\_DATETIME | DATE |
| 上次修改人 | PREV\_UPDATED\_BY | NUMBER(38) |
| 上次修改时间 | PREV\_UPDATED\_DATETIME | DATE |
| 最后修改人 | LAST\_UPDATED\_BY | NUMBER(38) |
| 最后修改时间 | LAST\_UPDATED\_DATETIME | DATE |
| 删除标识 | DEFUNCT\_IND | CHAR(1) |
| 默认标识 | DEFAULT\_IND | CHAR(1) |
| 有效标识 | VALID\_IND | CHAR(1) |
| 备注 | REMARKS | VARCHAR2(600) |
| 顺序号 | SEQ\_NO | NUMBER(3)或者NUMBER(8) |
| 金额 | AMOUNT | NUMBER(15,4)或者NUMBER(15,2) |
| 数量 | QTY | NUMBER(15,4) |
| 审核人 | AUDITED\_BY | NUMBER(38) |
| 审核时间 | AUDITED\_DATETIME | DATE |
| 退款原因 | REFUND\_REASON | VARCHAR2(100) |

## 5.2系统中固定的数值类型及长度：

* + - ID ：NUMBER(38)
    - 流水号：VARCHAR2(20)
    - 日期/时间：DATE
    - 标识：CHAR(1) 例如：删除标识、已使用标识、已过期标识等，且要有默认值Y/N
    - 金额：NUMBER(15,4), NUMBER(15,2)
    - 数量：NUMBER(15,4)
    - 顺序号:NUMBER(3)或者NUMBER(8)
    - 税率：NUMBER(15,6)
    - 百分比：>0 <100 比如：八折，数据库记录数字为80

## 5.3 his中常用词汇

|  |  |  |
| --- | --- | --- |
| **字段** | **命名** | **备注** |
| **账务模块** | | |
| 费用明细 | PATIENTACCOUNTTXN | PAT |
| 账单 | BILL |  |
| 发票 | RECEIPT |  |
| 发票分类 | RECEIPT\_CAT | 门诊发票、住院发票 |
| 发票类型 | RECEIPT\_TYPE | 发票、收据 |
| 折扣 | DISCOUNT |  |
| 收银机 | COLLECTION |  |
| 预交金 | DEPOSIT |  |
| 储值金 | STAKEHOLDER |  |
| 赠送金 | GIFTDEPOSIT |  |
| 个人支付金额 | PERSONAL\_AMOUNT |  |
| 会员 | CUSTOMER |  |
| 会员身份 | CUSTOMERCLASS |  |
| 会员服务 | CUSTOMERSERVICE |  |
| 会员卡 | MEMBERCARD |  |
| 实体套餐 | PACKAGE |  |
| 优惠券/代金券 | COUPON |  |
| 组织机构 | ORGANISATION |  |
| 付款 | PAYMENT |  |
| 支付方式 | PAYMANET\_MODE |  |
| 税率 | TAX\_RATE |  |
| **ADT模块** | | |
| 预约 | APPIONTMENT |  |
| 门诊预约 | REGNAPPOINTMENT | 缩写为REGN |
| 资源 | RESOURCE |  |
| 排班 | RESOURCESCHEME |  |
| 疫苗剂次 | VACCINEDOSE |  |
| 疫苗 | VACCINE |  |
| 患者 | PATIENT |  |
| 号别 | REGN\_TYPE |  |
| 挂号流水 | VISITREGNTYPE |  |
| 时段 | SESSION |  |
| 出院 | DISCHARGE |  |
| 入院 | ADMISSION |  |
| 住院 | IP |  |
| 门诊 | OP |  |
| 留观 | OBSERVATION |  |
| 接诊 | ACCEPT |  |
| 黑名单 | BLACKLIST |  |
| **QU队列** | | |
| 叫号 | CALLING |  |
| 队列 | QUEUE |  |
| 入队 | ENQUEUE |  |
| **OM模块** | | |
| 门诊 | OP |  |
| 住院 | IP |  |
| 医嘱 | ORDERENTRY | 缩写为OE |
| 检查 | EXAMINATIONREQUISITION |  |
| 检验 | LABREQUISITION |  |
| 手术 | OPERATION | 缩写为OT |
| 术后登记 | OTNOTES |  |
| 医嘱状态 | OST |  |
| 用法 | ROA |  |
| 医嘱数量 | ORDERED\_QTY |  |
| 医嘱数量单位 | ORDERED\_QTY\_UOM |  |
| 剂量数量 | DOSAGE\_QTY |  |
| 剂量单位 | DOSAGE\_UNIT |  |
| 周期数量 | DURATION\_QTY |  |
| 周期单位 | DURATION\_UOM |  |
| 处方号 | FORM\_NO |  |
| 频次 | FREQUENCY |  |
| 自备药 | SELF\_PREPARED |  |
| 医嘱确认 | ORDERCONFIRMATION | 缩写为OC |
| 医嘱执行 | PLANNEDEXECUTION | 缩写为PE |
| 医嘱执行记录 | ORDERSTATUSJOURNAL | 缩写为OSJ |
| 发药 | DRUGDISPENSE | 缩写为DD |
| 检验申请 | LAB\_REQUISITION |  |
| 检验结果 | LIS\_RESULT | lis传输的检验结果 |
| 检查申请 | EXAMINATIONREQUSITION |  |
| **体检模块** | | |
| 体检 | PES | 所有体检表结构的统一前缀 |
| 分检 | SUBSPECIALTYEXAM | 缩写为PSE，含义本质为科室检查 |
| 主检 | MAINEXAM | HIS中含义为在总检医生之前对分检结果进行初步整合，并非必要流程 |
| 总检 | TOTALEXAM | 体检总结结果并输出体检报告的过程 |
| 体检预约 | PES\_REGNBOOKING | 缩写为PRB，记录一次预约动作 |
| 体检预约人 | PES\_REGNBOOKINGPATIENT | 缩写为PRBP，记录一次预约动作预约的所有人 |
| 体检登记 | PES\_REGISTRATION | 缩写为PR，记录一次登记动作 |
| 体检登记人 | PES\_REGISTRATIONPATIENT | 缩写为PRP，记录一次登记动作下的所有登记人 |
| 套餐卡 | PACKAGECARD |  |
| 套餐 | PANEL |  |
| 项目 | ITEM |  |
| 子项目 | SUBITEM |  |
| 体检组 | PHYSICAL\_GROUP |  |
| **其他** | | |
| 报表 | RPT | 当前使用的报表模块下的所有表的前缀（除某些老表） |
| 即时消息 | INSTANTMESSAGE |  |